

版权注意事项：1、书籍版权归著者和出版社所有；  
2、本PDF仅用于个人获取知识，进行私底下知识交流；  
3、PDF获得者不得在互联网以任何目的进行传播；  
如有需要，请尽量购买正版实体书！支持书籍作者！！



从数据、基础平台、分析方法、行业应用4个维度，以场景化方式讲解数据从获取、预处理、挖掘、建模、结论分析与展现到系统应用的流程，以及机器学习的重要技术  
三位金融领域的大数据专家近10年行业实战经验总结，包含大量行业解决方案和案例，并公开源代码



技术丛书



Analytics and Applications with Business Cases  
Guide to Big Data & Machine Learning

# 大数据与机器学习

## 实践方法与行业案例

陈春宝 阙子扬 钟飞◎著



机械工业出版社  
China Machine Press

## 内容简介

---

三位资深大数据专家近10年经验总结，多位银行、互联网金融与Fintech公司大咖联袂推荐。本书立足商业实践，结合典型业务场景，详细阐述数据从获取、预处理、挖掘、建模、结论分析与展现到系统应用的整个流程。就完整性而言，覆盖数据、平台、分析和应用等企业内数据流转的主要环节；就内容而言，抛弃了理论与公式的堆积以及小明式的人造案例，选取大量翔实的案例展现数据从线下分析到线上应用的企业实战过程；就写作手法而言，力求兼顾实用主义和理论深度，用浅显的语言介绍复杂的分析应用过程，从实战角度诠释理论技术和算法的具体应用；就布局而言，按照数据与平台篇、分析篇和应用篇分别撰写。

数据与平台篇（第1~3章）立足找到数据、整合数据、使用数据三个角度，介绍数据在企业内的产生、存储、处理到分析、应用的闭环流过程，有助于数据工程师站在应用角度了解数据治理方法和重点：数据架构师可从中找到构建数据平台的指导思想和产品选型建议。

分析篇（第4~11章）选取企业实际案例，阐述数据是如何解决业务问题并产生价值的，帮助数据分析师掌握常用的数据挖掘与机器学习算法以及可视化技巧，从中找到分析灵感。

应用篇（第12~15章）选取标签系统、自助营销、个性化推荐和社会关系网络等当前最热门的大数据应用案例，介绍数据分析结论和模型的应用部署，帮助业务运营专家和管理者了解如何构建数据驱动的应用，让数据“自动”流转于各个环节。



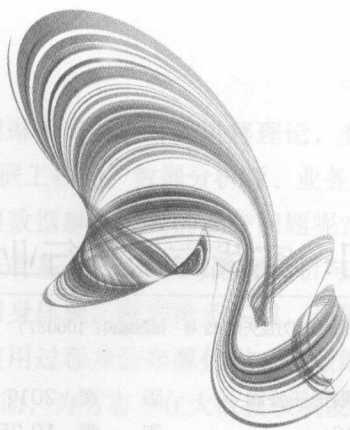
技术丛书

Analytics and Applications with Business Cases  
Guide to Big Data & Machine Learning

# 大数据与机器学习

## 实践方法与行业案例

陈春宝 阙子扬 钟飞◎著



### 本书的创作初衷

大数据方面的书籍可谓琳琅满目，有的介绍理论，有的介绍方法，有的传播理念。但是，大数据从业人员（如数据工程师、业务分析师、算法设计师等）应该掌握哪些知识与技能，如何应用数据？恐怕最能给出答案的还是实际的数据从业者。为此，我们结合多年在大数据领域的实践经验，站在企业实际应用的视角，介绍数据分析应用过程，并辅以大量行业案例，形成本书。本书力求做到理论与实践相结合，既可作为大数据分析应用过程的参考，也可作为大数据分析应用案例的参考。

### 本书特点

本书有三大特点：其一，本书从大数据分析应用的角度，对数据应用从业务需求、数据准备、数据分析、数据建模、数据应用等全流程进行了详细讲解；其二，本书结合大量行业案例，站在数据从业者的角度，对大数据分析应用进行了详细讲解；其三，本书结合大量行业案例，站在数据从业者的角度，对大数据分析应用进行了详细讲解。



机械工业出版社  
China Machine Press



## 图书在版编目 (CIP) 数据

大数据与机器学习：实践方法与行业案例 / 陈春宝，阙子扬，钟飞著. —北京：机械工业出版社，2017.1

(大数据技术丛书)

ISBN 978-7-111-55680-0

I. 大… II. ① 陈… ② 阙… ③ 钟… III. ① 数据处理 ② 机器学习 IV. ① TP274  
② TP181

中国版本图书馆 CIP 数据核字 (2016) 第 323978 号

## 大数据与机器学习：实践方法与行业案例

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：李 艺

印 刷：北京文昌阁彩色印刷有限责任公司

版 次：2017 年 1 月第 1 版第 1 次印刷

开 本：186mm×240mm 1/16

印 张：19.25

书 号：ISBN 978-7-111-55680-0

定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：(010) 88379426 88361066

投稿热线：(010) 88379604

购书热线：(010) 68326294 88379649 68995259

读者信箱：hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问：北京大成律师事务所 韩光 / 邹晓东

## Preface 前言

不畏浮云遮望眼，只缘身在最高层。

——王安石《登飞来峰》

数据科学家 = 统计学家 + 程序员 + 讲故事的人 + 艺术家

——Shlomo Aragon

## 本书的创作初衷

大数据方面的书籍可谓琳琅满目，有的讲解理论，有的介绍方法，有的传播理念。但是，大数据从业人员（如数据工程师、数据分析师、业务分析师、算法设计师等）应该掌握哪些知识与技能，如何应用数据解决现实的业务问题呢？恐怕最能给出答案的还是实际的数据从业者。为此，三位作者基于近 10 年的数据分析与应用经验，融合各自在商业银行、互联网金融和电商领域的切身体验，寓理论于实战，选取多个详实的案例，站在企业实际应用的角度介绍数据分析应用过程并公布源代码，并最终形成本书。本书对于读者开展数据分析工作能够提供直接帮助，为有志于在大数据领域发展的读者启航。

## 本书特点

本书有三大特点。其一，内容全面，覆盖大数据生态中的数据、基础平台、分析方法和应用四个领域，对数据应用从业务需求、数据准备、数据分析、挖掘建模、演示报告、成果应用等全流程进行了详细阐述；其二，以业务场景为主线，精选银行和互联网方面最具代表性的案例，站在数据消费者和分析师的角度，身临其境地介绍了数据如何产生价值，

## 图书在版编目(CIP)数据

寓理论于实战,让读者能知其所以然;其三,写作手法上遵循大道至简原则,用浅显的语言介绍复杂的数据分析应用过程,归纳数据分析师乃至数据科学家应该修炼的要点,既关注技术细节,又不拖泥带水,能为读者提供直接帮助。

ISBN 978-7-111-65580-0

## 本书定位

本书既可作为数据分析与商业分析人员的入门指引和案头工具,亦可为统计学、计算机科学、市场营销等专业研究生拓宽视野。

## 源代码下载

对书中源代码感兴趣的读者,可与作者联系(邮箱:64346837@qq.com)。

《数据科学》王安王——

陈永吉 + 人力资源管理 + 员电群 + 案例计划 + 案例计划

nomsg7A omoln2——

## 本书定位

1. 本书定位:本书是一本入门级的书籍,主要面向初学者,旨在帮助他们了解数据科学的基本概念和方法。本书内容涵盖了数据科学的基本概念、数据收集、数据清洗、数据分析和数据可视化等方面。本书采用通俗易懂的语言,结合大量的实例和图表,帮助读者理解数据科学的原理和应用。本书可作为数据科学入门教材,也可作为相关领域的参考书籍。

2. 本书特点:本书具有以下特点: (1) 内容全面:本书涵盖了数据科学的基本概念和方法,从数据收集到数据分析和数据可视化,形成了一个完整的知识体系。 (2) 通俗易懂:本书采用通俗易懂的语言,结合大量的实例和图表,帮助读者理解数据科学的原理和应用。 (3) 实用性强:本书提供了大量的实例和代码,帮助读者将理论知识应用到实际工作中。 (4) 适合初学者:本书是一本入门级的书籍,适合初学者阅读。 (5) 可作为教材:本书可作为数据科学入门教材,也可作为相关领域的参考书籍。

3. 本书结构:本书共分为10章,主要内容包括:第1章 数据科学概述;第2章 数据收集;第3章 数据清洗;第4章 数据分析;第5章 数据可视化;第6章 数据建模;第7章 数据评估;第8章 数据应用;第9章 数据伦理;第10章 数据科学未来展望。

4. 本书定价:本书定价为49.00元。

5. 本书出版:本书由机械工业出版社出版。

6. 本书发行:本书由机械工业出版社发行。

7. 本书印刷:本书由机械工业出版社印刷。

## 点读本书

本书是一本入门级的书籍,主要面向初学者,旨在帮助他们了解数据科学的基本概念和方法。本书内容涵盖了数据科学的基本概念、数据收集、数据清洗、数据分析和数据可视化等方面。本书采用通俗易懂的语言,结合大量的实例和图表,帮助读者理解数据科学的原理和应用。本书可作为数据科学入门教材,也可作为相关领域的参考书籍。



## Contents 目 录

## 前言

## 第一部分 数据与平台篇

## 第 1 章 数据与数据平台 ..... 3

## 1.1 数据的基本形态 ..... 4

## 1.1.1 数据环境与数据形态 ..... 4

## 1.1.2 生产数据 ..... 5

## 1.1.3 原始数据 ..... 5

## 1.1.4 分析数据 ..... 6

## 1.2 数据平台 ..... 7

## 1.2.1 数据仓库平台 ..... 9

## 1.2.2 大数据平台 ..... 13

## 1.2.3 MPP 数据库 ..... 22

## 1.2.4 NoSQL 数据库 ..... 23

## 1.3 应用系统 ..... 24

## 1.4 本章小结 ..... 25

## 第 2 章 数据体系 ..... 26

## 2.1 数据闭环 ..... 27

## 2.2 数据缓冲区 ..... 28

## 2.2.1 系统解耦 ..... 29

## 2.2.2 批量导出 ..... 31

## 2.2.3 FTP 传输 ..... 40

## 2.2.4 批量导入 ..... 42

## 2.3 ETL ..... 49

## 2.3.1 ETL 工具 ..... 50

## 2.3.2 ETL 作业 ..... 52

## 2.4 作业调度 ..... 56

## 2.5 监控和预警 ..... 56

## 2.5.1 使用监控工具进行监控 ..... 57

## 2.5.2 使用 BI 工具进行监控 ..... 57

## 2.6 本章小结 ..... 57

## 第 3 章 实战：打造数据闭环 ..... 59

## 3.1 数据缓冲区的基本规则 ..... 60

## 3.1.1 文件存储规则 ..... 61

## 3.1.2 文件命名规则 ..... 61

## 3.1.3 文件清理规则 ..... 62

## 3.2 自动加载的流程 ..... 62

## 3.2.1 扫描文件 ..... 63

## 3.2.2 下载文件 ..... 64

3.2.3	解压文件 .....	65
3.2.4	加载文件 .....	65
3.3	自动加载程序的数据库设计 .....	66
3.3.1	数据文件信息表 .....	67
3.3.2	数据文件状态表 .....	68
3.3.3	加载配置信息表 .....	69
3.3.4	数据缓冲区信息表 .....	70
3.3.5	目标服务器表 .....	70
3.4	自动加载程序的多线程实现 .....	71
3.4.1	ScanFiles .....	72
3.4.2	DownLoadAndUnZip .....	75
3.4.3	LoadToHive .....	77
3.4.4	LoadToOracle .....	78
3.4.5	自动加载程序的部署架构 .....	79
3.4.6	程序的维护和优化 .....	80
3.5	本章小结 .....	80

## 第二部分 分 析 篇

第 4 章	数据预处理 .....	83
4.1	数据表的预处理 .....	84
4.2	变量的预处理 .....	85
4.2.1	缺失值的处理 .....	85
4.2.2	极值的处理 .....	90
4.3	变量的设计 .....	91
4.3.1	暴力衍生 .....	91
4.3.2	交叉升维 .....	92
4.4	变量筛选 .....	95
4.4.1	筛选显著变量 .....	95
4.4.2	剔除共线性 .....	96
4.5	本章小结 .....	100

## 第 5 章 聚类, 简单易用的客户细分方法 .....

5.1	从客户细分说起 .....	102
5.1.1	为什么要做客户细分 .....	102
5.1.2	怎么做客户细分 .....	103
5.1.3	聚类分析, 无监督的客户 细分方法 .....	107
5.2	谱系聚类 .....	107
5.2.1	基本步骤 .....	107
5.2.2	案例: 公司客户差异化 服务 .....	110
5.2.3	谱系聚类方法的题外话 .....	115
5.3	K-means 算法 .....	116
5.3.1	基本步骤 .....	116
5.3.2	案例: 电商卖家细分 .....	117
5.3.3	K-means 算法的题外话 .....	121
5.4	本章小结 .....	121

## 第 6 章 关联规则挖掘, 发现产品 加载和交叉销售机会 .....

6.1	销售的真谛: 让客户买得更多 .....	123
6.1.1	案例: 电商的生意经 .....	123
6.1.2	案例: 富国银行的 “商店”经营模式 .....	124
6.1.3	案例总结 .....	125
6.2	交叉销售 .....	126
6.2.1	为什么要做交叉销售 .....	126
6.2.2	怎么做交叉销售 .....	126
6.3	关联规则挖掘, 发现交叉销售 机会 .....	128
6.3.1	Apriori 算法 .....	129



6.3.2	Apriori 算法的主要指标	129
6.3.3	Apriori 算法的基本步骤	131
6.4	案例: 信用卡产品交叉销售	131
6.4.1	准备数据	132
6.4.2	SAS 实现	132
6.4.3	结果分析	133
6.4.4	序列关联分析	136
6.4.5	结果应用	137
6.5	本章小结	138

## 第 7 章 社交网络分析, 从“关系” 的角度分析问题

7.1	先看几张美轮美奂的图片	140
7.2	社交网络分析方法	142
7.2.1	定义	142
7.2.2	应用场景	142
7.2.3	网络识别算法	143
7.3	案例: 电商通过订单数据识别 供应链	144
7.3.1	供应链及供应链金融	144
7.3.2	识别核心企业及其上下游 关系	144
7.3.3	分析结果的业务应用	149
7.4	案例: P2P 投资风险防范	151
7.4.1	案例背景	151
7.4.2	防范方法	152
7.5	本章小结	153

## 第 8 章 线性回归, 预测客户价值

8.1	数值预测	156
8.2	回归与拟合	157
8.2.1	回归就是拟合	157

8.2.2	在 Excel 中添加趋势线预测	158
8.3	案例: 信用卡客户价值预测	159
8.3.1	确定预测目标	159
8.3.2	准备建模数据	161
8.3.3	模型拟合	163
8.3.4	模型评估	165
8.4	基于客户价值分层的业务策略	167
8.5	本章小结	167

## 第 9 章 Logistic 回归, 精准营销的 主要支撑算法

9.1	大数据时代的精准营销	170
9.1.1	精准营销	170
9.1.2	基于大数据的精准营销 模式	171
9.1.3	如何做到精准	172
9.2	Logistic 回归算法介绍	173
9.2.1	算法原理	173
9.2.2	关键步骤	174
9.3	案例: 信用卡消费信贷产品的 精准营销	176
9.3.1	案例背景	176
9.3.2	数据准备	176
9.3.3	数据预处理	180
9.3.4	建模	182
9.3.5	模型评估	185
9.4	预测模型的应用与评估	189
9.5	本章小结	189

## 第 10 章 决策树类算法, 反欺诈 模型“专家”

10.1	决策树, 重要的分类器	191
------	-------------	-----

10.2	决策树的关键思想	192
10.2.1	理财客户画像案例背景	192
10.2.2	关键思想一:递归划分	194
10.2.3	关键思想二:剪枝	197
10.3	案例:电商盗卡交易风险识别	198
10.3.1	案例背景	198
10.3.2	以 SAS 实现	199
10.3.3	以 Clementine 实现	201
10.3.4	以 R 实现	204
10.4	随机森林	208
10.5	本章小结	209

## 第 11 章 数据可视化,是分析更是设计

11.1	数据演示之道	210
11.1.1	好“色”之图	211
11.1.2	版式有形	212
11.1.3	数据发声	214
11.2	个性化地图	215
11.2.1	案例背景:存款增长率指标展示	215
11.2.2	获取地理位置的经纬度数据	216
11.2.3	定制地图背景和图标	217
11.2.4	生成地图	220
11.3	文本分析	222
11.3.1	案例:电商的客户评价分析	222
11.3.2	分词	223
11.3.3	词云制作	224
11.3.4	情感分析	225
11.4	本章小结	227

## 第三部分 应用篇

### 第 12 章 标签系统

12.1	认识标签系统	231
12.2	标签系统的设计	233
12.2.1	标签系统的层次结构	233
12.2.2	标签系统的更新规则	233
12.2.3	机器学习模型转化为标签	235
12.3	标签系统的实现	236
12.3.1	标签映射表	237
12.3.2	标签系统的前端实现	238
12.3.3	标签系统的数据后端实现	238
12.3.4	标签系统的在线接口实现	242
12.4	本章小结	242

### 第 13 章 数据自助营销平台

13.1	数据自助营销平台的价值所在	245
13.1.1	自动化营销,提升工作效率	245
13.1.2	降低营销成本,提升用户体验	247
13.1.3	个性化营销,提升响应率	248
13.1.4	统一管理,便于效果追踪	249
13.2	数据自助营销平台的实现原则	249
13.2.1	数据营销活动的节点	249
13.2.2	数据自助营销平台的基础:标签系统	251

13.2.3 数据自助营销平台的 批量任务 .....	252	14.3.1 系统框架 .....	275
13.2.4 实时数据营销 .....	254	14.3.2 推荐系统的刷新 .....	276
13.3 数据自助营销平台的场景实例 .....	254	14.3.3 部署一个可用的推荐 系统 .....	276
13.3.1 客户生命周期管理 .....	254	14.4 本章小结 .....	280
13.3.2 用卡激励计划 .....	257		
13.4 本章小结 .....	260		
<b>第 14 章 基于 Mahout 的个性化 推荐系统 .....</b>	<b>261</b>	<b>第 15 章 图计算与社会网络 .....</b>	<b>281</b>
14.1 Mahout 的推荐引擎 .....	262	15.1 社会网络和属性图 .....	282
14.1.1 Mahout 的安装配置 .....	262	15.2 Spark GraphX 与 Neo4j .....	283
14.1.2 Mahout 的使用方式 .....	263	15.2.1 Scala 编程语言 .....	284
14.1.3 协同过滤算法 .....	264	15.2.2 Cypher 查询语言 .....	285
14.1.4 Mahout 的推荐引擎 .....	265	15.3 使用 Spark GraphX 和 Neo4j 处理社会网络 .....	286
14.2 规模与效率 .....	268	15.3.1 背景说明 .....	286
14.2.1 Mahout 推荐算法的适用 范围 .....	268	15.3.2 数据准备 .....	286
14.2.2 通过分布式解决规模和 效率的问题 .....	270	15.3.3 Spark GraphX 处理原始 网络 .....	287
14.3 实现一个推荐系统 .....	275	15.3.4 Neo4j 交互式查询分析 .....	291
		15.3.5 更多的应用场景 .....	295
		15.4 本章小结 .....	296

第一部分 Part 1

# 数据与平台篇 (Data & Infrastructures)

合抱之木，生于毫末；九层之台，起于累土；千里之行，始于足下。

世界的本质是数。

迟序之数，非出神怪，有形可检，有数可推。

——祖冲之

数学是知识的工具，亦是其他知识工具的泉源。所有研究顺序和度量的科学均和数学有关。

——笛卡儿

可以通过数据认识自身：人类全身的肌肉大约有 630 块，由 60 亿条肌纤维构成，而起着重要作用的大脑则由 140 亿个细胞构成。

可以通过数据描述我们的工作：周一上午 10:00-11:30 召开会议，讨论公司第三季度的销售目标。

可以通过数据描述我们的行为：花费 6088 元购买一台 iPhone 6 手机，中速游泳 60 分钟消耗约 1000 千卡的热量。

还可以通过数据认识我们所处的环境：现在是 14:00，当前温度为 25℃，上个月 CPI 同比上涨 1.4%，蔬菜和水果价格上涨了 6.7%。

甚至可以通过数据认识遥不可及的物体：太阳直径为 1 392 000 公里，表面温度达 57 809 开尔文……数据让我们认识了世间万物，那么我们该如何认识数据本身呢？

数据的本质是一个十分深邃且宽泛的话题，甚至带有哲学的意味。作为技术类书籍，

对于大部分非计算机专业出身的分析人员和业务人员来说，数据库领域的专业术语简直让人抓狂，非要搞得那么高深吗？大可不必。

数据科学家是数据的应用者，以最大限度来提炼数据价值为目的，不必像数据仓库开发者那样对数据的存储、结构以及数据仓库的内生技术一清二楚，但应该站在找到数据、拼接数据、使用数据的角度，大体了解数据的分布、处理逻辑，以便为分析快速地准备素材。



## 第1章 Chapter 1

## 数据与数据平台

合抱之木，生于毫末；九层之台，起于垒土；千里之行，始于足下。

——《老子》

世界的本质是数。

——毕达哥拉斯

数据时时刻刻在伴随着我们的工作和生活，就像空气围绕着我们一样，以致于我们常常忽略了它的存在。但如果你立志做一个崇尚数据的人，静下心来像科学家研究空气一样研究数据，就会发现数据为我们认知事物打开了一条全新的途径。

可以通过数据认识自身：人类全身的肌肉大约有 639 块，由 60 亿条肌纤维构成，而起着重要作用的大脑则由 140 亿个细胞构成。

可以通过数据描述我们的工作：周一上午 10:00~11:30 召开会议，讨论公司第三季度的销售目标。

可以通过数据描述我们的行为：花费 6088 元购买一台 iPhone 6 手机，中速游泳 60 分钟消耗约 1000 千卡的热量。

还可以通过数据认识我们所处的环境：现在是 14:00，当前温度为 28℃，上个月 CPI 同比上涨 1.4%，蔬菜和水果价格上涨了 6.7%。

甚至可以通过数据认识遥不可及的物体：太阳直径为 1 392 000 公里，表面温度达 57 809 开尔文……数据让我们认识了世间万物，那么我们该如何认识数据本身呢？

数据的本质是一个十分深奥且宽泛的话题，甚至带有哲学的意味。作为技术类书籍，

本书不尝试从哲学的角度研究数据，而是基于实践，从思维和技术手段出发来认识、理解、处理并分析周围的数据。为了更加具体，本书研究的数据定位于企业经营数据。

本章首先将从数据的基本形态入手，介绍企业中数据的来源和表现形态；然后介绍与之相关的数据平台，并简单介绍两类应用系统。在着手处理数据之前，让我们先对数据有一个清晰的认识。

## 1.1 数据的基本形态

我们不是自然科学家，但是可以借鉴自然科学的思路来看待数据问题。问题是数据具有形态吗？虽然数据并不具有固态、液态或气态等形态，但是可以根据需要为数据定义属于自己的专属形态。

一旦为数据赋予了恰当的形态，并在一定范围内（比如在一个公司内部）达成共识，形成对数据的系统化认识，就可以基于这些数据形态提出相应的管理和使用方案，提升数据的效率和价值。

一般情况下，对于企业经营中产生的数据，可以定义为三种形态：生产数据、原始数据和分析数据。这些数据形态的产生，是基于企业应用系统所在的生产环境和分析环境而存在的，在深入讨论数据形态之前，我们先来熟悉一下数据所在的环境。

### 1.1.1 数据环境与数据形态

数据环境是指数据存储、处理、转换所处的物理环境，常见的数据环境有生产环境、分析环境和测试环境。

生产环境是生产应用系统实时运行所在的环境，而生产应用系统则是一系列业务逻辑的组合。我们可以把生产环境想象成人的身体，生产应用系统就是人体中的各个系统（消化系统、呼吸系统等），业务逻辑则是这些系统中的“经络”，而数据便是运行于经络之中的“气血”。数据从“经络”中的一个“穴位”流转 to 另一个“穴位”，并在“流淌”中发生变化，所以，生产环境中的数据是“动态变换”的数据，我们称为生产数据。

分析环境是与生产环境物理解耦的一个数据环境。在生产环境中，由于数据总是处于不停变化中，这些数据的变化将直接反映为业务逻辑结果的变化，因此不应该尝试在生产环境中对数据进行分析处理。为了不影响生产环境的正常运行，需要将生产环境中的“动态”数据的快照保存下来（例如每日凌晨将时间戳为昨日的数据导出），这些数据快照是“静态”的，我们称为分析数据，保存分析数据的物理环境即我们所说的分析环境。

在实际中，还有另外一个环境，即测试环境。测试环境中的数据也是独立于生产环境和分析环境的，由于测试环境的数据通常不是有效的数据，因此本书不关注测试环境的数据。

至此，根据数据所处的环境，我们将数据定义为三种基本形态：生产数据、原始数据和分析数据。

生产数据存在于生产环境之中，分析数据存在于分析环境之中。此外，在生产数据和分析数据之间，还存在一种过渡形态的数据，即原始数据。图 1-1 展示了数据环境及其对应的数据形态。

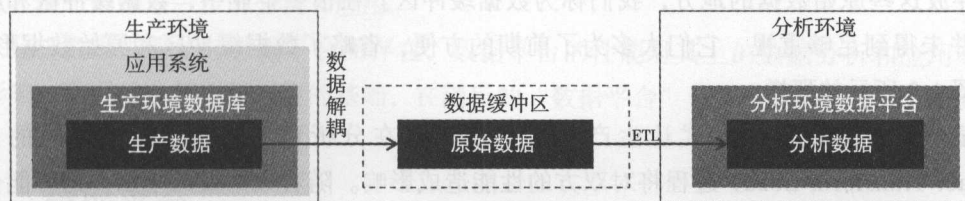


图 1-1 数据环境及其对应的数据形态

注意，图 1-1 中所示的原始数据，既不属于生产环境也不属于分析环境，这意味着它不直接用于生产，也不直接用于分析。原始数据作为生产数据到分析数据的中间形态存在，本书随后的章节将进一步讨论原始数据的相关问题。

### 1.1.2 生产数据

生产数据是应用系统中在线使用的数据，它可能是一个生产系统的生产环境数据库中的数据，比如在一个 P2P 借贷平台的系统中，用户进行注册、充值、投资等行为产生的数据将被记录到生产环境数据库中，这些数据即为生产数据。

生产数据是动态的，会随着业务应用的变化而变化，比如用户账户余额数据，会随着用户投资的变化而变化。任何存在于生产环境中的数据，都在时刻准备发生改变，只不过有些生产数据的变化频率特别低而已，比如用户的年龄信息。

正常情况下，数据分析师并不直接接触生产数据，但需要注意的是，有些生产数据是从分析数据而来的。比如用户标签数据，它本身是从分析数据构建的，属于分析数据。但这些标签数据一旦用于应用系统，例如作为推荐系统的底层数据，即转化为生产数据，这种情况下，应用系统输出结果的质量将受到分析数据的直接影响。

### 1.1.3 原始数据

由于生产数据是动态的数据，而过去大量的分析工具和分析方法很难处理动态改变的数据（流处理已经改变了这种情况）。为了在不影响生产应用系统的情况下分析和处理这些数据，我们需要将这些数据从生产系统解耦。

从生产系统解耦的数据即是原始数据。数据解耦的过程一般包括数据脱敏（如屏蔽电话



号码、去除住宅详细信息等)、信息筛选(抛弃不需要的字段)、批量导出(如在 T 日凌晨批量导出 T-1 日的交易明细数据)等。

原始数据可以以多种形式存在,例如存储在生产数据库备库中,或者以文本文件的格式存放在文件服务器中。无论以何种形式存在,原始数据都应该独立于生产环境和分析环境,这可以避免分析环境对生产环境的干扰。

存放这些原始数据的地方,我们称为数据缓冲区。在很多企业中,数据缓冲区和原始数据并未得到足够重视,它们大多为了前期的方便,省略了数据缓冲区和原始数据形态,就像图 1-2 所示的那样。

显然,数据直连的方式让生产环境直接暴露在分析环境之上,两者之间的 ETL (Extract-Transform-Load) 过程将对双方的性能造成影响。随着数据量的增加,这可能会带来数据管理和应用上的灾难。

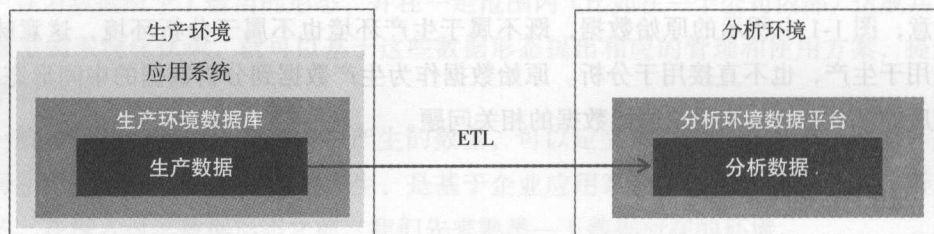


图 1-2 省略数据缓冲区的数据直连

本书极力推荐图 1-1 所示的方式,虽然它比图 1-2 要复杂,但在数据管理和可扩展性方面有非常大的优势。后面的第 2.1 节中会深入讨论该问题。

#### 1.1.4 分析数据

分析数据是对原始数据进行属性筛选、标准统一之后,使用优化存储的方式存放于分析环境中的数据。从原始数据到分析数据的关键步骤在于 ETL 过程。

比如,原始数据中的一张表 A 可能包含 100 个字段,经过 ETL 之后,得到了一个包含 45 个字段的表 B,其中的日期格式进行了统一,且滤除了一些特殊字符,并将表 B 存放于分析环境数据平台的关系数据库 Oracle 中。这样,原始数据中的表 A 完成了属性筛选和标准统一(日期格式),转换成了分析数据表 B。

另一种需要标准统一的情景根源于原始数据本身的多样性。由于原始数据来源于不同的生产应用系统,其数据格式及字段含义均存在差异。例如,原始数据存放的格式可能有 Windows 文本、Linux 文本、主机格式文本、数据库文件等多种形式;字段含义上的差异则更加多样,比如,由原始数据文件 A 中性别字段使用 1 表示男性、2 表示女性,而原始数

据文件 B 中性别字段使用 M 表示男性、F 表示女性。通过标准统一，可以约定所有的分析数据统一使用 1 表示男性、2 表示女性。数据统一可为数据分析和数据应用铺平道路。

经过 ETL 之后的分析数据，为了进一步提高存储效率和读取效率，需要使用技术手段进行存储优化，比如创建索引、进行分区、分表存储、使用大数据平台等。

通过对原始数据的提炼和优化，分析数据具有了信息集中、标准统一、分析效率高等特点，便于数据进一步的分析和应用。

分析数据需要依托数据平台而存在，数据平台的性能对其上的数据分析和应用有决定性影响。数据平台是分析环境的基础，在随后的“数据平台”章节中，我们将详细介绍。

## 1.2 数据平台

数据平台是存放分析数据的平台，也是支持大多数数据分析和数据挖掘应用的底层平台，它使用了统一的数据清洗与处理规则，因而可以保证从基础平台上输出的数据内容是一致的。

传统的数据平台基本等同于大家熟悉的“数据仓库”，但互联网浪潮让人们的数据采集、存储和应用提出了越来越高的要求，传统数据仓库平台独力难支，因此“现代化”的数据平台是多种数据库产品的融合。图 1-3 是一个精简化的现代数据平台架构图。

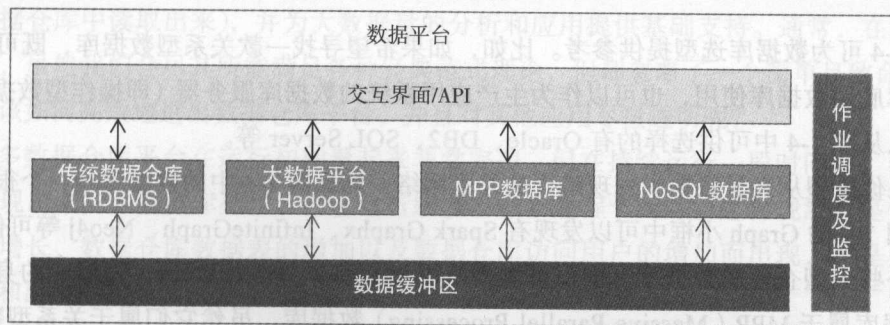


图 1-3 数据平台架构示意图

现代的数据平台融合了传统数据仓库、大数据平台、MPP 数据库、NoSQL 数据库等多种数据产品，这些数据库产品之间互为补充，组成统一的数据平台。

从传统的关系型数据库开始，数据库产品逐渐细分，这些细分产品在特定场景中比传统的关系型数据库表现出了更好的性能。图 1-4 展示了一些主流的数据库产品，注意到有很多数据库产品是“跨界”产品，例如，Oracle 同时属于关系型、分析型、操作型三类数据库。

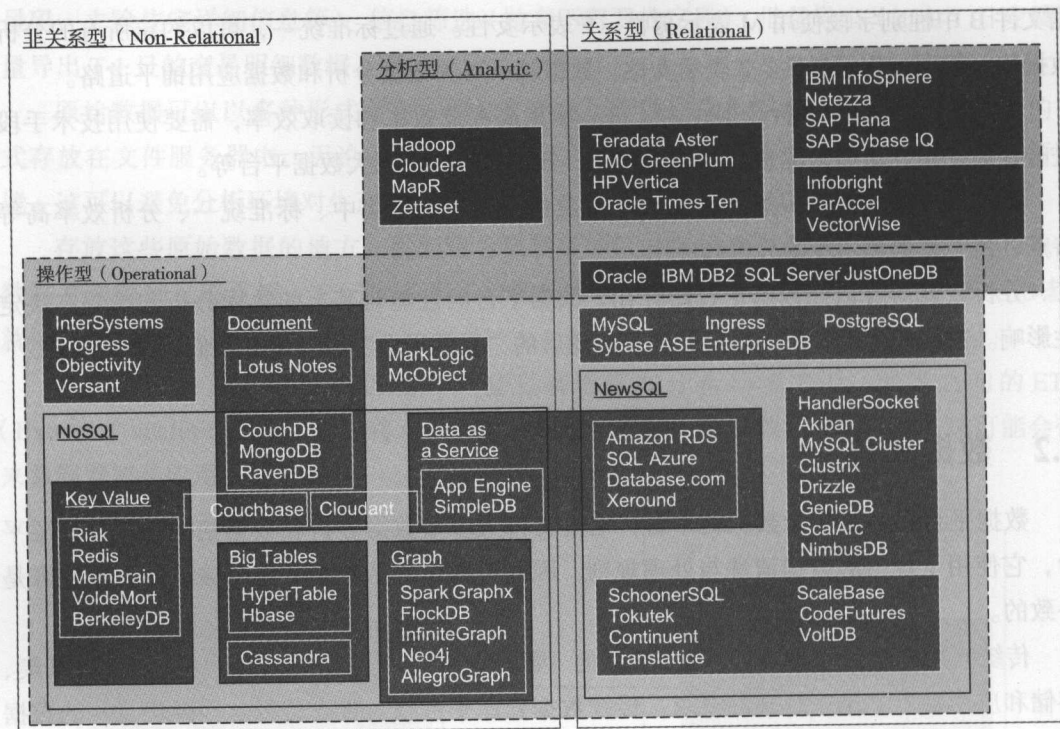


图 1-4 数据库产品图册

图 1-4 可为数据库选型提供参考。比如，如果希望寻找一款关系型数据库，既可以作为数据仓库底层数据库使用，也可以作为生产应用系统的数据库服务器（即操作型数据库）使用，那么从图 1-4 中可供选择的有 Oracle、DB2、SQL Server 等。

如果你希望从大量数据中发现隐含的关系网络，那么图 1-4 中的数据库是一个很好的选择，从图 1-4 的 Graph 小框中可以发现有 Spark Graphx、InfiniteGraph、Neo4j 等可供选择。

在一些大型企业的平台中，可能会出现 Teradata、GreenPlum、Vertica 的身影。这三种数据库属于 MPP（Massive Parallel Processing）数据库。虽然它们属于关系型数据库，但它们采用了一种与传统关系型数据库不同的存储方式，即列存储方式（Oracle、DB2、SQL Server 等使用行存储方式）。这种列存储方式在面对大规模数据时，能表现出更好的效率，比如创建列索引、并行处理、集群操作等。

MPP 数据库的问题在于其高昂的价格，这对于很多小型企业来说是一个“致命伤害”，目前国内使用该产品的企业一般为大型企业。第 1.2.4 节会对 MPP 数据库做进一步介绍。

NoSQL 数据库是伴随着互联网应用中崛起的新星。通过简化数据存取模式（相对 RDBMS 而言），减少了数据库管理系统（DBMS）的附加开销，专注于读/写效率的提升，非常适合对读取速度要求较高，且对数据不一致性有容忍的应用环境中。



归为 NewSQL 数据库之列的 SQL Azure 和 MySQL Cluster 是传统关系型数据库的云计算版本。目前该类数据库在国内企业的应用中尚不常见,但随着云计算的普及,预计国内一些大型企业有可能选用 NewSQL 数据库。

下面,我们对图 1-3 中组成数据平台的各主要模块分别进行介绍。

### 1.2.1 数据仓库平台

传统数据仓库平台,大多是基于关系型数据库搭建的。在 Windows 服务器平台上,一般选用 SQL Server、Oracle、DB2、MySQL 数据库中的一种或者两种混合搭建。

在 Linux 服务器平台上,可选择的数据库有 DB2、Oracle、MySQL (虽然 SQL Server 正在准备发布 Linux 版本)。数据仓库平台的建模过程已经有了完善的理论基础,并且目前市面上也有很多相关书籍,限于篇幅,本书不做详细介绍。数据仓库工程师可以参阅以下书籍。

□《数据仓库》(《Building The Data Warehouse》),机械工业出版社。

□《数据库系统:数据库与数据仓库导论》(《Database Systems: Introduction to Databases and Data Warehouses》),机械工业出版社。

#### 数据入口和出口

数据仓库本质上是解决大批量数据的入口和出口问题(简单来说,即数据写入数据仓库和从数据仓库中读取出来),并为大数据量的分析和应用提供基础支持。通常,在数据仓库设计时,虽然我们会尽其所能满足各种经典范式理论,但却忽略了一个简单且致命的问题:数据应该如何高效地进出数据仓库平台,并且对其他应用尽量透明呢?

很多数据仓库平台在运行初期看起来非常完美,但在持续运行一段时间后,各种问题逐渐显现出来,如数据库死锁、作业运行缓慢、ETL 过程卡死等。这种现象总是伴随着数据量的增长、数据仓库数据表的增加以及数据仓库访问用户的增加而出现,这是典型的数据入口和出口问题顽疾。

一个能够长期稳定提供“顺滑”数据服务的数据仓库才是一个“好”的数据仓库,而不仅仅是看这个数据仓库在设计时采用了什么数据库产品、满足了多少范式理论。

根据笔者的经验,“好”的数据仓库的入口和出口至少要关注两条规则,即数据的更新规则和存储规则。更新规则会影响数据的入口效率和出口效率,存储规则主要影响数据的出口效率。

#### (1) 数据的更新规则

实际场景中,一个数据库中的数据表总是面临数据加载(Load)的问题。比如一张交易明细表,每天凌晨需要将昨日的交易明细数据插入进去;而一张账户信息表,则需要将每天新增的账户信息插入及将状态改变的账户信息更新。

归纳起来，数据的更新规则分为两种：增量更新和全量更新。上面的交易明细表即是增量更新方式，每天将新增的数据插入；而账户信息表则可以采用全量更新的方式，每天将最新的账户信息表从数据源重新加载进来，并覆盖原有数据。图 1-5 展示了这两种更新方式。

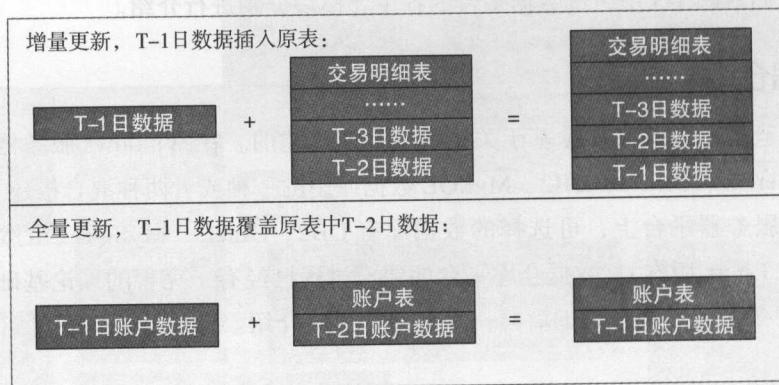


图 1-5 增量更新与全量更新

需要明确的是，在数据仓库中进行的数据更新一般是大数据量的数据更新，这与生产应用系统中的单条数据更新在处理方式上有很大的不同。在数据仓库场景中你将会面临百万（条）或者千万（条）级别的数据量，如果使用 SQL 中的 Insert（或 update）语句，则 Insert 语句带来的事务管理、日志记录等会严重降低数据库性能，并且整个插入过程的耗时巨大，导致数据表长时间不可用。

通常在面临大批量数据更新时，正确的做法是使用批量导入命令进行批量导入操作，并尽可能避免 Update 操作。我们将在第 2 章详细介绍批量导入命令。

但是使用批量导入命令面临的主要问题是“部分提交”的问题，因为数据库管理系统（DBMS）通常不将批量导入命令置于 DBMS 的事务控制范畴之中，这意味着整个批量导入过程并不验证数据库范式，也不记录操作日志，因此，如果批量导入中途出现异常，那么之前导入的数据将无法回滚。

比如，一个信用卡中心每天的交易明细数据有 900 万条，在往交易明细表中批量导入该批数据的时候，由于网络问题导致数据库连接中断，此时 500 万条记录已经导入交易明细表中且无法回滚，这就出现了“部分提交”问题。

这种情况是一个让人郁闷的问题，因为必须将导入的 500 万条数据删除（delete），然后重新运行批量导入命令。而从一个庞大的数据表中删除 500 万条数据简直是一种灾难，相信经历过的人肯定不想再来一次。

当全量更新时，除了“部分提交”的问题，还可能面临“数据断档”的问题。因为在全量更新时，首先需要将原表中的数据清空，之后再新数据批量导入至清空后的表中。

从数据清空到新数据全部导入，这个过程即为“数据断档”，因为在这个过程中数据表是无法对外提供服务的。

“部分提交”和“数据断档”是数据仓库运行过程中经常遇到的问题。一种可行的方案是增加中间表，即数据先完整导入中间表，然后再从中间表插入目标表，如图 1-6 所示。

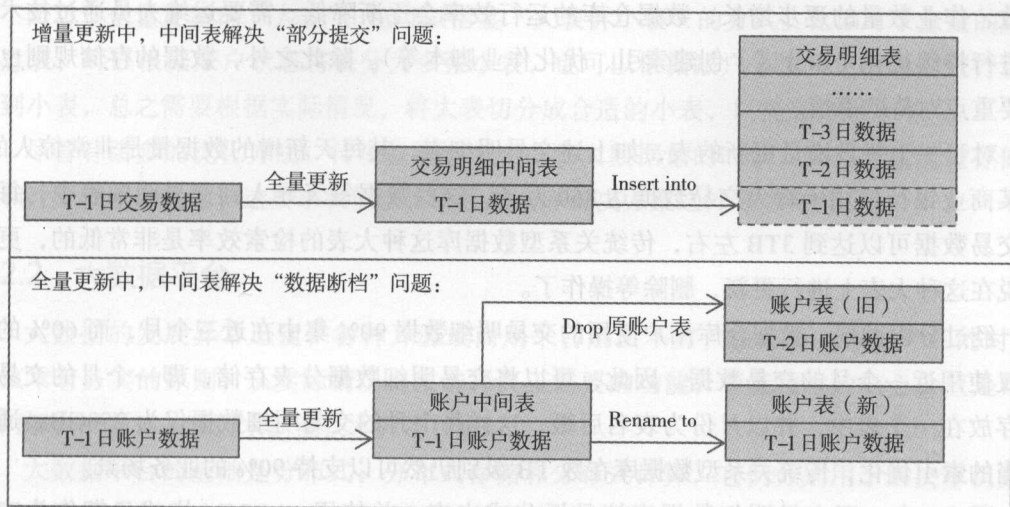


图 1-6 通过中间表进行批量加载

在图 1-6 中，原来交易明细表的增量更新拆分为以下两步。

- 1) 清空“交易明细中间表”，并将 T-1 日交易数据全量更新至“交易明细中间表”中。
- 2) 将“交易明细中间表”中的数据全部 insert into 到“交易明细表”中。

上述两个步骤中，假设步骤 1) 出现问题，则清空“交易明细中间表”后重新执行步骤 1)；而如果步骤 2) 出现问题，则数据库通过自动回滚会将已经 Insert 的数据回滚掉，不会出现数据不完整的情况。

在图 1-6 中，对于全量更新的账户表，拆分为以下三步完成。

- 1) 清空“账户中间表”，并将 T-1 日账户数据全量更新至“账户中间表”。
- 2) Drop 掉原账户表 (T-2 日账户数据)。
- 3) 将“账户中间表”重命名为“账户表”，实现数据切换。

上述三个步骤，规避了“部分提交”的问题，同时在数据加载至中间表的过程中，原“账户信息表 (T-2 日数据)”仍可以正常提供数据服务 (只不过数据并不是那么“新鲜”)，因此解决了数据空档期的问题。

通过中间表，上述的增量更新、全量更新出现问题后，可以简单地通过重新执行解决，因此可以方便地通过程序自动完成上述过程。这种自动化过程规避了人工操作风险，降低了数据仓库维护成本，使数据仓库更加“顺滑”。第 3 章将详细介绍如何采用这种理念实现



数据的增量更新和全量更新。

(2) 数据的存储规则

数据仓库在运行初期一般都表现良好，这是因为运行初期的数据量、用户数、作业数量等均比较少，数据服务器本身的硬件性能可足以支撑这些业务。但是，随着数据量、用户数、作业数量的逐步增长，数据仓库的运行效率会逐渐降低。需要运维人员通过技术手段进行持续优化（如分区、创建索引、优化作业脚本等）。除此之外，数据的存储规则也是需要重点对待的事情。

对于一个每日增量更新的表，如上述交易明细表，其每天新增的数据量是非常惊人的，如某商业银行信用卡每天交易数据达 860 万条，这些数据每天导入到交易明细表中，每年的交易数据可以达到 3TB 左右，传统关系型数据库这种大表的检索效率是非常低的，更不用说在这种大表上进行更新、删除等操作了。

经过分析发现，数据仓库用户使用的交易明细数据 90% 集中在近三个月，而 60% 的作业仅使用近一个月的交易数据。因此，可以将交易明细数据分表存储，即一个月的交易数据存放在一个表中，并以月份为表名后缀，这样每个月的交易明细数据仅为 250GB，通过适当的索引优化，传统关系型数据库在数 TB 级别仍然可以支持 90% 的业务场景。

图 1-7 中，原交易明细数据表按月拆分成小表，并使用 yyyyMM 格式日期作为表名后缀。拆分成小表后，如果是访问最近一个月的交易明细数据，则直接访问表 `trx_dtl_YYYYMM` 即可；而如果想访问最近三个月的交易数据，则通过视图 `v_trx_dtl_r3m` 访问即可。这样，在存储空间几乎不变的情况下，大幅度提高读取效率，从而使整个数据仓库的出口变得“顺滑”；另外，按月存储的交易明细表由于体积小，能更方便进行优化管理（如创建索引、迁移数据等），间接提高了数据仓库的入口效率。

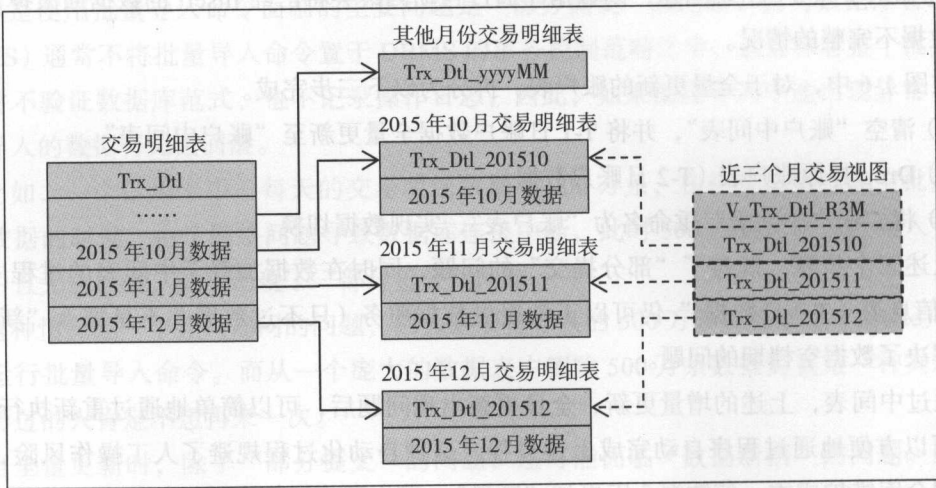


图 1-7 分表存储

拆分成小表后，由于表的命名规则固定（原表名+月份后缀），因此整个拆分表以及组合成视图的过程均可以通过脚本作业自动完成，提高了管理效率且节省了人力成本。

总体来说，在面对增量更新的“大表”时，数据的存储规则是“大表拆小表，小表组视图”，基本依据在于小表的访问效率高于大表的。

而对于全量更新的“大表”，也可以借鉴“大表拆小表”的方式。比如对于全量的账户信息表，可以根据账户状态将表分为多张小表，也可以根据账户号按一定的规则进行切分得到小表，总之需要根据实际情况，将大表切分成合适的小表，以提高访问效率。

尽管有诸多的优化方法，传统数据仓库在面临大数据量的时候，仍然无法规避存储空间和计算效率的问题，这要求我们在传统数据仓库解决方案之外寻求突破。

### 1.2.2 大数据平台

大数据的发展异常迅猛，各种开源或商用平台层出不穷，在选择大数据产品的时候，应该本着务实的原则，从实际情况出发选择真正需要的功能，毕竟技术的核心价值是帮助我们解决问题，一味追求新潮技术并不可取。

大数据平台的基础是分布式：分布式存储和分布式计算，它们分别用于解决单机数据库面临的两大困境，即数据量的问题和计算效率的问题，如图 1-8 所示。单机数据库由于无法支持分布式，所以其存储容量和计算能力的瓶颈难以突破，而大数据平台通过分布式扩展轻易解决了这两个问题。

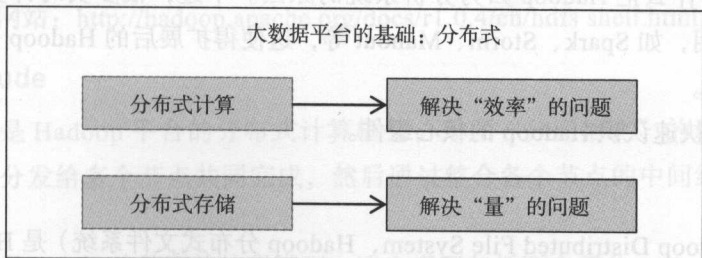


图 1-8 大数据平台的基础：分布式

分布式与扩展性密不可分，当存储和计算能力不足时，显而易见的方案就是增加集群中的机器，在存储价格和 CPU 价格日益下降而人力成本日益上涨的今天，这比从优化传统数据库系统着手要简单高效，且成本更低。这也是为什么说“能通过增加机器解决的问题都不是问题”的原因。

目前实际应用中的大数据平台基本是以开源的 Hadoop 平台为核心，不同厂商在此基础上进行封装和扩展，形成自己的产品线。比较知名的商用大数据平台产品有：Cloudera 的 CDH，华为的 FusionInsight。



图 1-9 为大数据平台的核心组件。其底层基础模块 HDFS (Hadoop Distribute File System, Hadoop 分布式文件系统) 用于提供分布式存储能力; MapReduce 分布式计算框架用于提供分布式计算能力。HDFS 可支持多种组件, 包括 Hive 数据仓库、数据挖掘、流处理等, 其外围的 Zookeeper 负责集群之间的协作管理, 使众多的机器可以成为统一的集群, Flume 和 Sqoop 则是大数据平台数据的入口和出口, 负责与其他系统的数据交互。

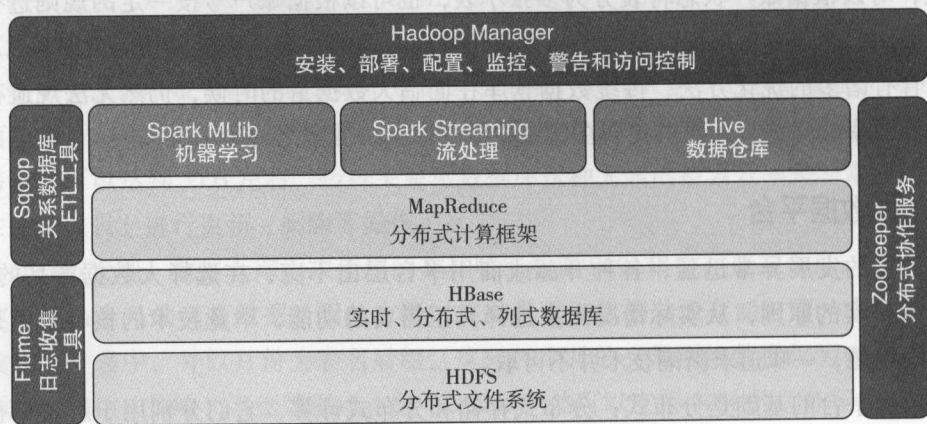


图 1-9 大数据平台的核心组件

需要注意的是, Hadoop 的核心模块提供的是离线、批量的计算, 本身并不适合强实时环境, 这也是为什么把 Hadoop 归为分析系统的原因。不过, 很多实时计算的组件经常与 Hadoop 结合使用, 如 Spark、Storm、Mahout 等, 这使得扩展后的 Hadoop 平台具备了一定的实时处理能力。

下面让我们快速认识 Hadoop 的核心组件。

## 1. HDFS

HDFS (Hadoop Distributed File System, Hadoop 分布式文件系统) 是 Hadoop 平台的文件基础, 就如 Windows 环境的 NTFS (New Technology File System)、Linux 环境的 ext 文件系统 (Extended File System) 一样。

你可能并不需要关心 HDFS 的技术细节, 但需要了解 HDFS 的主要设计理念是针对超大文件存储 (几百 MB~PB 级别), 一次写入、多次读取。它不适合用在低延迟的场景, 也不适合存储大量小文件, 这是因为 HDFS 文件的元数据 (文件基本信息, 如文件名称、路径、存放的 DataNode 节点信息等) 是存放在 NameNode 的内存中的, 大量的小文件会消耗 NameNode 的内存, 而且会影响 MapReduce 对文件的处理效率。

建立在 HDFS 之上的 Hive 数据仓库也是针对大数据量的数据分析工具的, 在数据量未达到一定规模时, Hive 并不能体现出效率优势 (在小数据量时, Hive 的效率远低于传统关

系型数据库的)。这里有一个经验值, 当一个表中的数据达到数百 GB 的时候, 使用关系型数据库进行读 / 写、Join、Sum、Group By 等操作时会耗费大量时间, 运行一段 SQL 脚本可能需要数个小时, 这时 Hive 将体现出绝对的优势。

可以使用 Hadoop Shell 命令进行 HDFS 文件的交互式操作。Hadoop Shell 命令与 Linux 的 Shell 命令非常相似, 多数情况下只需要在 Linux Shell 命令前加上 “hadoop fs” 即可。常用的命令列举如表 1-1 所示。

表 1-1 Hadoop Shell 常用命令

Hadoop Shell 命令	说 明
hadoop fs -cat URI	将路径指定文件的内容输出到 stdout
hadoop fs -cp URI [URI ...] <dest>	将文件从源路径复制到目标路径。这个命令允许有多个源路径, 此时目标路径必须是一个目录
hadoop fs -copyFromLocal <localsrc> URI	从本地文件系统中复制单个或多个源路径到 HDFS 文件系统
hadoop fs -copyToLocal URI <localdst>	从 HDFS 文件系统中复制单个或多个源路径到本地文件系统
hadoop fs -rm URI [URI ...]	删除指定的文件。只删除非空目录和文件
hadoop fs -rmr URI [URI ...]	递归版本
hadoop fs -test [-ezd] URI	-e 检查文件是否存在。如果存在, 则返回 0 -z 检查文件是否是 0 字节。如果是, 则返回 0 -d 如果路径是一个目录, 则返回 1; 否则返回 0

表 1-1 中的这些命令可以方便我们操作 HDFS 中的文件, 第 3 章将使用 Java 调用 Hadoop Shell 命令的方式实现多线程批量导入数据至 Hive 数据仓库。详细的 Hadoop Shell 命令可访问官方网站: <http://hadoop.apache.org/docs/r1.0.4/cn/hdfs shell.html>。

## 2. MapReduce

MapReduce 是 Hadoop 平台的分布式计算框架, 采用“分而治之”的思想, 把对大规模数据集的操作分发给多个节点共同完成, 然后通过整合各个节点的中间结果, 得到最终结果。

MapReduce 是一个简单易用的编程模型, 这个模型包括两个部分: Map 过程和 Reduce 过程, 由 Map 和 Reduce 两个函数分别实现。

Map 函数接受一个键-值对, 经过处理产生一组中间键-值对。MapReduce 框架会将 Map 函数产生的键-值对里键相同的值传递给一个 Reduce 函数。

Reduce 函数接受一个键及相关的一组值, 将这组值进行合并产生一组规模更小的值(通常只有一个或没有值)。

深入了解 MapReduce 的实现方式, 对理解大数据平台的设计理念很有帮助, 虽然类 SQL 语言 Hive 已经代办了 MapReduce 的大部分工作, 但是本书建议读者能够理解 MapReduce 的原理和简单实现, 这样对于深入理解大数据大有帮助。

关于 MapReduce 的更多内容可以参考《Hadoop 权威指南（第3版）》（《Hadoop: The Definitive Guide, 3rd Edition》），清华大学出版社。

### 3. Hbase

Hbase 是基于 HDFS 的列式存储分布式数据库，属于 NoSQL 中的 Big Tables 范畴（见图 1-4）。Hbase 号称“能够提供高可靠性、高性能、列存储、可伸缩、实时读/写的数据库系统”，因此在一些时效要求相对较高的场景，会出现 Hbase 应用的身影。

但是需要注意的是，Hbase 是基于 HDFS 文件存储的，而 HDFS 并不是针对低延时的场景设计的，因此 Hbase 本身的实时性能并不高，所以它比较适用于“异步的、准实时、高维度”的场景中。在后面第三部分中介绍的“实时数据营销平台”即使用了 Hbase 作为数据存储。

如果你已经对 Hbase 比较了解，则可以跳过本节下面的内容。

#### （1）Hbase 的概念视图与物理视图

通过概念视图和物理视图，可以帮助我们快速理解 Hbase 的列存储模式。首先看一下 Hbase 的概念视图，如图 1-10 所示。

Row Key	Time Stamp	Column Family finance	Column Family status
Charles	T10		status:weight = "78"
Charles	T8		status:height = "180"
Charles	T5	finance:balance = "99.99"	
Charles	T3	finance:balance = "88.88"	
Charles	T2	finance:balance = "66.66"	

图 1-10 Hbase 的概念视图

从图 1-10 中可以看到 Hbase 的几个关键概念，即 Row Key、Time Stamp 和 Column Family。

Row Key 是 Hbase 的标识行字段，相同的 Row Key 在数据逻辑上属于同一行，图 1-10 中的数据均属于 Row Key=“Charles”这一行。

Time Stamp 是 Hbase 的时间戳，在写入数据时自动记录。因此 Hbase 可以自动记录列的历史版本，在需要保存历史变动记录的场景里，这个特征非常有用。图 1-10 中的 finance:balance 列即有三个版本 T5、T3、T2，在读取数据时，Hbase 默认读取最新版本。

Column Family 是 Hbase 的列族，Hbase 按照列族组织物理存储。每个列族可以随意增加列，因此 Hbase 的列一般表示为“列族名称:列名称”方式。图 1-10 中列族 finance 含有



一个列 html，写作 `finance:balance`；列族 `status` 含有两个列，分别是 `weight` 和 `height`。

在图 1-10 的概念视图里，尽管表可以看成是一个稀疏的行的集合，但在物理上，它是按列族分列存储的。图 1-11 是 Hbase 的物理视图。

列族 `status` 的物理视图：

Row Key	Time Stamp	ColumnFamily status
Charles	T10	status:weight = "78"
Charles	T8	status:height = "180"

列族 `finance` 的物理视图：

Row Key	Time Stamp	ColumnFamily finance
Charles	T5	finance:balance = "99.99"
Charles	T3	finance:balance = "88.88"
Charles	T2	finance:balance = "66.66"

图 1-11 Hbase 的物理视图

注意到图 1-10 中的空白格在物理上是不存储的，因为根本没有必要存储。因此，若要获取 T8 时间的 `finance:balance`，结果就是空；同样，若获取 T5 时间的 `status:height`，结果也是空。

## (2) Hbase 的读 / 写操作

Hbase 在实际应用中一般用作准实时分布式数据库，在数据量较小的时候表现并不突出，但在数据量巨大时的点写入和点查询性能均高于关系型数据库的。

Hbase 的另一个优势在于，可以几乎无限制地进行列的扩展，这非常适用于替代传统关系数据库的“大宽表”，因此用于客户标签体系（客户会拥有成千上万个的标签，并且是一个非常稀疏的数据表）的存储表是非常合适的，在后面第三部分的介绍中，将会使用 Hbase 作为用户标签系统的数据表。

Hbase 表可以通过 Hbase shell 进行交互读 / 写。代码清单 1-1 的 Hbase Shell 脚本创建了一个 Hbase 表，并向其中插入了数据，然后读取了相关信息。

更多关于 Hbase Shell 命令的使用方法可参考 Hbase 官网。

代码清单 1-1

```
# 创建表 clt_tag，两个列族，即 base_info 和 trx_info
```

```

create 'clt_tag','base_info','trx_info'
# 往表 clt_tag 中插入数据, rowkey=10001, 列族 base_info 增加一个列 name, 值为 queziyang
put 'clt_tag','10001','base_info:name','queziyang'
# 往表 clt_tag 中插入数据, rowkey=10001, 列族 base_info 增加一个列 age, 值为 31
put 'clt_tag','10001','base_info:age','31'
# 往表 clt_tag 中插入数据, rowkey=10001, 列族 trx_info 增加一个列 total_amt, 值为 1500.3
put 'clt_tag','10001','trx_info:total_amt','1500.3'
# 读取 clt_tag 中 rowkey=10001、列 base_info:name 的数据
get 'clt_tag','10001','base_info:name'

```

### (3) Hbase 的批量读 / 写

Hbase 适用于点读 / 写场景, 在进行大批量数据操作的时候会面临一些问题。同时, 存在这样一种情况: Hbase 用于点查询数据库, 但是数据本身需要定期更新, 例如每天晚上需要将数千万条数据更新至 Hbase 表中, 以便第二天进行查询。在后面第三部分介绍的实时客户标签系统中, 就需要每天将更新后的用户标签批量导入至 Hbase 表中, 因此 Hbase 面临批量更新的问题。

由于 Hbase 的更新等同于插入, 因此批量更新方式即批量导入的方式。比较高效的批量更新操作, 一般通过 MapReduce 程序直接生成 Hbase 存储文件 HFile, 然后将生成的 HFile 加载到 Hbase 表中。具体参阅第 2 章批量导入 Hbase 的章节。

关于 Hbase 的更详细内容, 可以参考《HBase 权威指南》(《HBase: The Definitive Guide》), 人民邮电出版社。

## 4. Hive

Hive 是 Hadoop 平台的数据仓库工具, 它将 HDFS 文件直接映射为数据表, 并提供类 SQL (Hive SQL, HQL) 语句进行数据表操作。HQL 在执行时转化为 MapReduce 作业, 在实际的数据批量操作场景中, Hive 可以完成绝大部分本来需要 MapReduce 程序完成的任务, 因此 Hive 的出现降低了 MapReduce 的使用成本, 仅仅通过撰写简单的 HQL 语句就可以享受到 MapReduce 的强大功能, 如图 1-12 所示。Hive 是使用 Hadoop 大数据平台数据仓库的必备技能。

Hive 作为大数据平台数据仓库工具, 同传统数据仓库平台一样需要有统一的设计原则。在设计大数据平台数据仓库系统时, 可以借鉴传统数据仓库的设计理念, 把 Hive 表看成是关系型数据库的表。

同传统数据仓库一样, Hive 数据仓库中表的数据加载同样分为增量更新和全量更新的情况, 但由于 Hive 本身的分布式特点, 其数据表的存放规则与传统数据仓库的有所不同。

### (1) Hive 分区表与增量更新

Hive 数据仓库与传统关系型数据仓库一样, 也需要解决数据的出口与入口问题, 不过由于 Hive 数据本身基于 HDFS 分布式存储, 因此我们更关注 Hive 数据仓库制定数据的更

新规则。回忆关系型数据仓库的更新规则：增量更新与全量更新，这两条规则也可以用于 Hive 数据仓库。

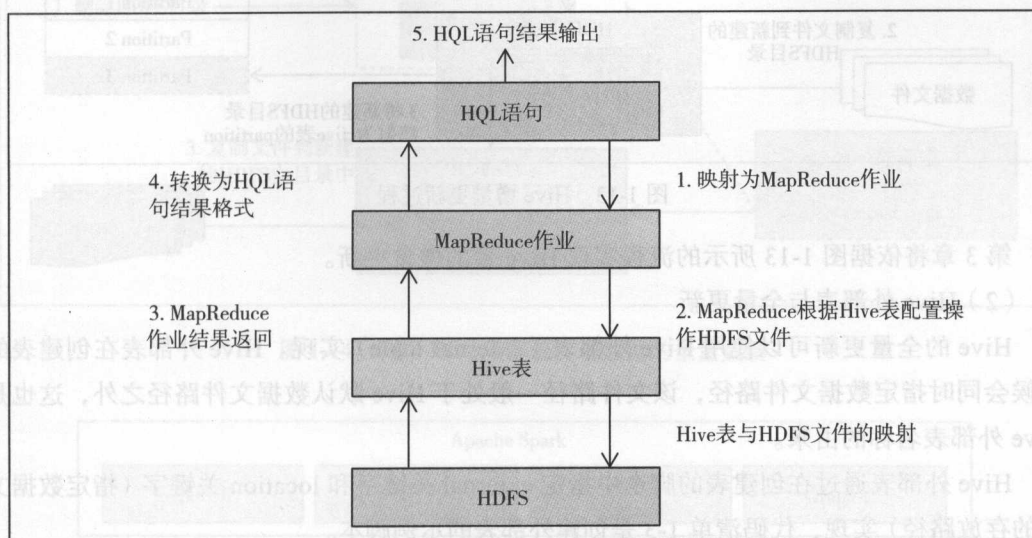


图 1-12 HQL 转为 MapReduce 作业

增量更新可以使用 Hive 的分区表来“完美”解决。Hive 的分区表类似于关系数据库的表分区，即根据分区键将数据分散在不同的分区（partition），通过并行读取提高效率。Hive 的分区表可以方便指定每个分区的 HDFS 路径，因此可以通过程序自动完成，代码清单 1-2 给出的脚本作为示例创建了一个分区表，并指定了 load\_day 作为分区键。

代码清单 1-2

```

create table adobe_log_app(
  id      int,
  name    string,
  age     int,
  tel     string
)
partitioned by(load_day string)
row format delimited
fields terminated by '\t'
stored as textfile;

alter table adobe_log_app add partition (load_day='20150927') location
  '/hive/data/adobe_log_app/20150927';
  
```

“alter”命令往表 adobe\_log\_app 中增加了一个分区，并通过 location 指定了该分区的 HDFS 路径。显然，通过程序只需要将表名、分区键值作为参数传入，即可以实现增量数据的自动加载，其过程如图 1-13 所示。

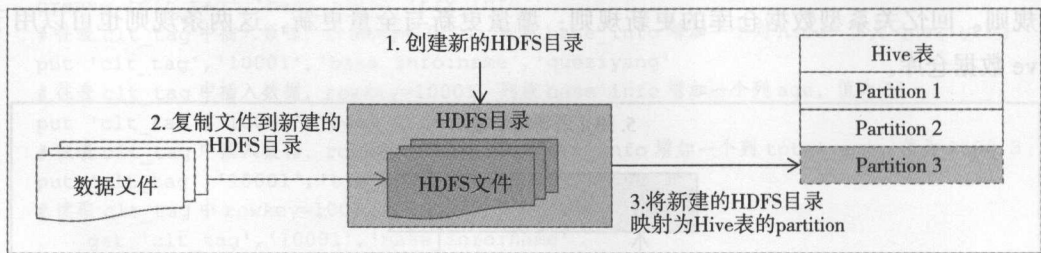


图 1-13 Hive 增量更新过程

第 3 章将依据图 1-13 所示的流程实现 Hive 表的增量更新。

### (2) Hive 外部表与全量更新

Hive 的全量更新可以使用 Hive 外部表 (external table) 实现。Hive 外部表在创建表的时候会同时指定数据文件路径, 该文件路径一般处于 Hive 默认数据文件路径之外, 这也是 Hive 外部表名称的由来。

Hive 外部表通过在创建表的脚本中指定 external 关键字和 location 关键字 (指定数据文件的存放路径) 实现, 代码清单 1-3 是创建外部表的示例脚本。

代码清单 1-3

```
create external table clt_act_info(  
    act_id      bigint,  
    act_typ     string,  
    limit       int,  
    ...  
)  
row format delimited  
fields terminated by '\t'  
stored as textfile  
location '/hive/data/clt_act_info';
```

Hive 外部表进行全量更新时, 分为三个步骤: ①删除原数据所在的 HDFS 目录 (location 所指目录); ②创建新的 HDFS 目录 (目录的路径与原目录保持一致); ③将新数据文件复制到新的 HDFS 目录。图 1-14 展示了其整个过程。

## 5. Spark

Spark 是基于内存的类 MapReduce 通用并行框架, 它拥有 MapReduce 所具有的优点, 并且抛弃了 MapReduce 的文件中转功能, 不需要读/写 HDFS, 因此 Spark 能更好地适用于数据挖掘与机器学习等需要多次迭代的计算场景。

Spark 包含四大主要组件, 即 Spark SQL、Spark Streaming、MLlib (Machine Learning)、GraphX, 如图 1-15 所示。



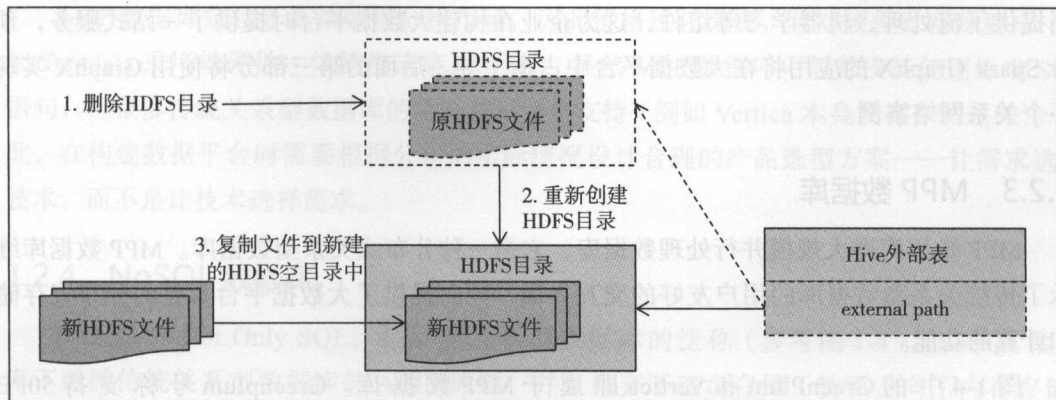


图 1-14 利用 Hive 外部表实现数据的全量更新

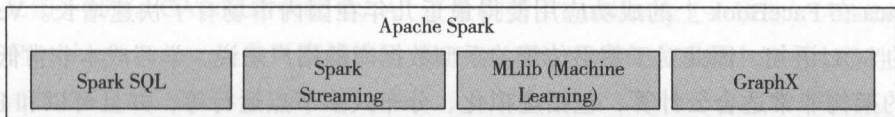


图 1-15 Spark 的主要组件

Spark SQL 同 Hive SQL 一样，都是类 SQL 语言，Spark SQL 是基于 Spark 进行分布式计算的，而 Hive SQL 是基于 MapReduce 进行分布式计算的。由于 Spark 计算框架是内存实现，因此理论上 Spark SQL 的速度要比 Hive SQL 的快，当然其对内存的要求也比 Hive SQL 的高。

Spark Streaming 是 Spark 的流数据处理组件，它将流式计算分解成一系列短小的批处理作业，也就是把输入数据按照 batch size（如 1 秒）分成一段一段的数据，对每一小段的数据进行处理，整个过程可以看成是一连串非常小的批处理过程。Streaming 流处理一般使用消息队列（如 Kafka）作为输入端，实时收取消息队列中订阅的消息并进行处理。Spark Streaming 能够与 Kafka 完美对接，有兴趣的读者可以参阅官网文档。

MLlib 是 Spark 的机器学习组件，它提供协同过滤、回归、聚类、分类、人工神经网络等主流机器学习算法，这些算法都提供了 API 接口，可以方便通过 Java 或 Python 进行编程调用，因此非常容易结合大数据平台使用。

GraphX 是 Spark 的图计算和图挖掘引擎，图计算在目前的关系网络分析中受到越来越多的重视。关系网络涉及多个主体之间的复杂联系，如果使用关系型数据库或者主流的数据分析工具（如 SAS、R、Python）实现对整个关系网络的描述，已经是一件非常困难的事情，更不用说进一步的分析和计算了。Graphx 融合了图并行以及数据并行的优势，虽然与单纯的计算机段的性能相比不如 GraphLab 等的计算框架，但是如果从整个图处理流水线的视角（图构建、图合并以及最终结果的查询）看，那么性能就具有明显的优势。另外，由于 Spark



还提供了流处理、机器学习等组件，这为企业在构建大数据平台时提供了一站式服务，预计 Spark GraphX 的应用将在大数据平台中占据主导。后面的第三部分将使用 GraphX 实现一个关系网络案例。

### 1.2.3 MPP 数据库

MPP 数据库即大规模并行处理数据库，它是一种分布式关系型数据库。MPP 数据库继承了传统关系型数据库的用户友好的交互界面，同时提供了大数据平台具有的分布式存储和计算的功能。

图 1-4 中的 GreenPlum 和 Vertica 即属于 MPP 数据库。Greenplum 号称支持 50PB (1PB=1000TB) 级海量数据的存储功能，目前国内的大众点评、阿里巴巴、华泰保险、中国远洋等均使用了该产品。

Vertica 在 FaceBook 上的成功应用使得最近几年在国内市场有了快速增长。Vertica 使用标准的 SQL 语句，因此对于熟悉传统关系型数据库的用户来说，学习成本非常低。另外 Vertica 的架构非常适合云计算，包括虚拟化、分布式多节点运行等，并且可以和 Hadoop/MapReduce 集成，因此非常利于市场推广，目前国内的招商银行已经引入该产品。

MPP 数据库在架构上可以分为 Master-Slave 架构（这其实也是 Hadoop 的架构）和 Share-Nothing 架构（无共享节点架构）两种，如图 1-16 所示。GreenPlum 属于 Master-Slave 架构，但其随后的产品可能转换为无共享节点架构，Vertica 属于无共享节点架构。

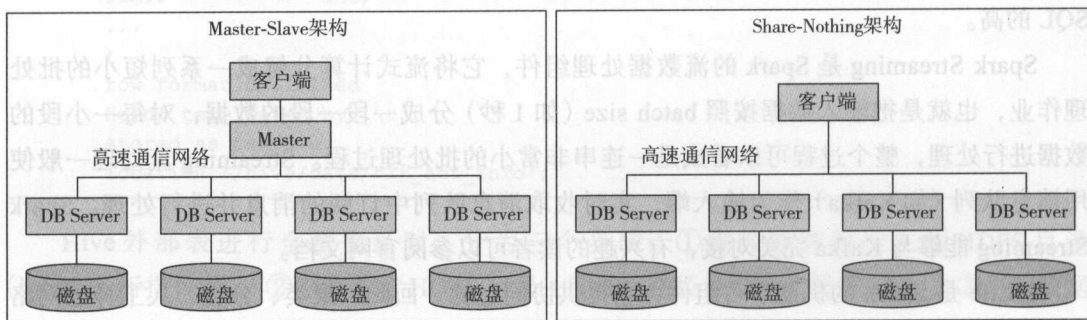


图 1-16 MPP 的两种架构

在 Master-Slave 架构中，Master 有可能成为系统瓶颈，但是 Master 节点本身并不负责计算，仅用于 Slave 节点之间的控制以及交互数据的转发，因此在实际使用中，Master 成为瓶颈的场景并不常见。

在无共享节点架构中，各个处理单元都有自己私有的 CPU、内存、硬盘等资源，不存在共享资源，各节点之间通过协议进行通信，各节点独自处理自己的数据，处理后的结果可能向上层汇总或在节点间流转。

MPP 数据库定位于高端数据分析市场，因此价格比较昂贵，且对硬件有特殊要求（例如 Teradata 采用软硬件一体销售策略）。另外，虽然 MPP 数据库一般都支持使用标准 SQL 语句，但很多传统关系型数据库的功能并不完全支持（例如 Vertica 本身没有存储过程）。因此，在构建数据平台时需要根据公司的实际情况设计合理的产品选型方案——让需求选择技术，而不是让技术选择需求。

### 1.2.4 NoSQL 数据库

NoSQL 即 Not Only SQL，是对非关系型数据库的泛称（参考图 1-4）。NoSQL 数据库不遵循传统关系型数据库的 ACID 原则，并且抛弃了磁盘存储，转而走向了内存存储。NoSQL 数据库大多应用于分布式应用系统中。

相对于传统关系型数据库的 ACID 理论，NoSQL 理论基础主要基于 CAP 原则（也叫 CAP 定理，见图 1-17）。CAP 定理中的 C、A、P 分别指 Consistency（一致性）、Availability（可用性）、Partition tolerance（分区容错性）。NoSQL 理论对分布式系统中的三个特性进行了如下归纳。

- 1) 一致性 (C)。一致性被称为原子对象，任何的读/写都应该看起来是“原子”的。写后面的读一定能读到前面写的内容，所有的读/写请求都好像被全局排序。
- 2) 可用性 (A)。对任何非失败节点都应该在有限时间内给出请求的回应（请求的可终止性）。
- 3) 分区容错性 (P)。允许节点之间丢失任意多的消息，当网络分区发生时，节点之间的消息可能会完全丢失。

CAP 定理由 Eric Brewer 教授提出，并由 Lynch 等人于 2002 年证明了 Brewer 的猜想。CAP 定理告诉我们，一个分布式系统不可能同时满足一致性、可用性和分区容错性这三个需求，最多只能同时满足两个。

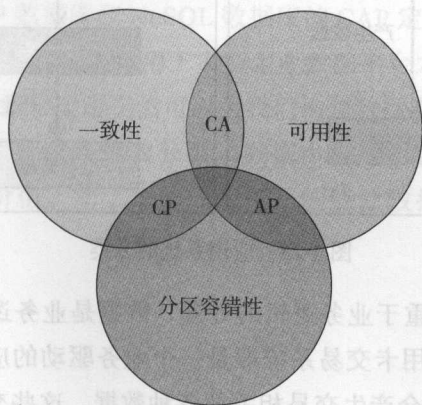


图 1-17 CAP 定理

根据 CAP 定理, 可以根据不同的应用场景选取其中的两个作为设计方向。目前, NoSQL 数据库作为实时应用系统数据库一般遵循 AP 原则。满足 AP 原则的 NoSQL 数据库一般采用 Key-Value 内存数据库, 如 Redis (参考图 1-4)。

但 CAP 定理正在面临诸多质疑。CAP 的概念定义比较模糊, 而且 CAP 没有考虑不同的基础架构、不同的应用场景、不同的网络基础和用户需求, 而 C、A、P 在这些不同场景中的含义可能完全不同, 这种无差异化的定义直接导致了概念的不明确, 同时也成为 CAP 被质疑的源头。

不过, CAP 定理仍然有显著的指导意义, 它至少告诉我们在设计分布式系统和选择 NoSQL 数据库产品时需要考虑的基本方向, 它还提醒我们分布式系统与传统系统架构存在的差异性, 你必须根据实际的应用慎重选择分布式框架。

### 1.3 应用系统

前面已经为数据定义了三种基本形态, 并认识了各种数据平台。从图 1-1 中还可以看到, 原始数据主要是由应用系统产生的。作为数据的源头, 我们有必要从数据的角度重新认识应用系统。

从数据角度看, 应用系统可以分为两类: 业务驱动的应用系统和数据驱动的应用系统, 如图 1-18 所示。

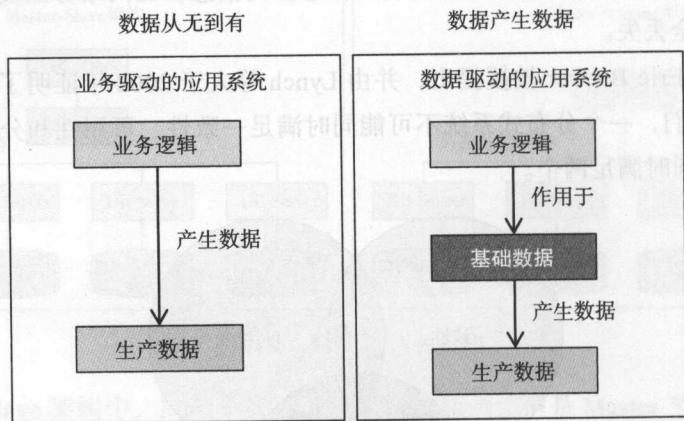


图 1-18 应用系统的分类

业务驱动的应用系统侧重于业务逻辑的处理, 数据是业务逻辑运行的直接结果, 它不依赖于现有数据。例如, 信用卡交易系统即是一个业务驱动的应用系统, 持卡人每一次刷卡消费 (触发业务逻辑), 都会产生交易相关的各种数据, 这些交易数据的产生是一个从无

到有的过程。

数据驱动的应用系统，其主要特点在于业务逻辑需要作用于基础数据，才能产生新的数据。数据驱动的应用系统一般需要与数据模型、数据标签等结合使用，这些数据模型及数据标签均基于已有的历史数据构建而成。数据模型通常作为业务逻辑的一部分，为业务逻辑提供决策支持。典型的数据驱动的应用系统如个性化推荐系统、数据营销系统等。

概括来说，业务驱动的应用系统是数据从无到有的过程，数据驱动的应用系统是数据产生数据的过程。无论是业务驱动的应用系统还是数据驱动的应用系统，其产生的数据经过数据脱敏、数据解耦后才能成为原始数据。

数据要产生价值，归根结底要体现在应用系统中。国内有很多企业在数据离线应用中做得很好，包括数据的分析、数据模型等，但这些数据在系统化、自动化的过程中产生了严重的滞后，这显然是数据向价值转换路上的一个不足。后面第三部分的内容正是着眼于将数据的离线应用推广到在线应用，将离线创建的数据模型和推荐模型系统化、自动化，从而更好地实现数据价值。

## 1.4 本章小结

首先，本章介绍了数据的基本形态以及与之相关的各种数据平台，从数据分析和应用角度来看，数据的基本形态包括生产数据、原始数据和分析数据三种，它们分别对应于三种环境，即生产环境、数据缓冲区和分析环境。

其次，本章着重介绍了分析环境的数据平台，包括传统数据仓库平台和大数据平台。在数据仓库平台中介绍了数据的更新规则和存储规则，这是数据仓库平台解决数据入口和出口问题的重要方法。大数据平台主要介绍了大数据平台的基本组件，目的在于给读者一个整体概念。

然后，本章介绍了 MPP 数据库和 NoSQL 数据库的 CAP 定理。MPP 数据库是一种定位于高端分析市场的数据库产品，一般应用于大型企业数据平台之中；NoSQL 数据库的 CAP 定理虽然备受争议，但是了解该定理仍然可以为我们构建数据应用提供参考。

最后，本章简单介绍了应用系统的分类。数据由应用系统而来，最后仍然需要应用到系统中去，才能转化为价值。第 2 章的内容正是围绕让数据更好地产生价值这一主题展开的。



## 数据体系

迟序之数，非出神怪，有形可检，有数可推。事类相推，各有攸归，故枝条虽分而同本干知，发其一端而已。又所析理以辞，解体用图，庶亦约而能周，通而不黜，览之者思过半矣。

——刘徽《九章算术注》

管理就是决策。

——赫伯特·西蒙

数据存在于生产环境、数据缓冲区以及分析环境的各个节点中，并且由各种技术手段支撑着数据的存储和计算。通常，在企业中，生产环境由开发部门负责，而分析环境和数据缓冲区则由数据部门负责，物理环境分离以及管理上的隔离会让人们产生一种错觉：数据是数据部门的事情，应用系统是开发部门的事情。这对数据的应用是非常不利的。

我们应该试图从更高层次上来对待数据，要打破管理和认知上的壁垒，就要让数据像金融系统中的资本那样运转起来。隔离的、静止的数据是乏味的，就如货币一样，需要流动才能增值。

数据的流动伴随着形态的变化（回忆数据的三种形态：生产数据、原始数据、分析数据），我们知道数据最终仍然要回归于生产系统（从生产中来，到生产中去），一切离开了生产应用的数据分析和处理都是徒劳无益的。

因此，要构建一个健康的数据体系，这个体系要像货币流通系统那样能够循环和增值，这是本章将要讨论的主题。

## 2.1 数据闭环

基于数据流动的理念，我们想象一个完美的数据闭环：数据在三种形态之间的循环转换，从生产系统产生，经过整个闭环后，最终仍然应用于生产系统。在这个数据闭环中，数据形态的每次转化，都伴随着数据的相应增值，如图 2-1 所示。

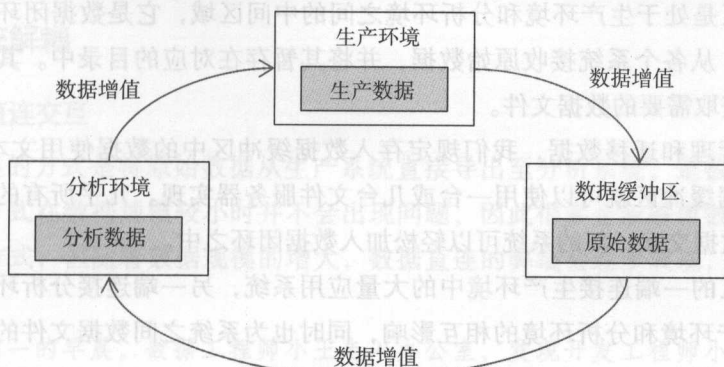


图 2-1 数据闭环

处于数据闭环中的分析环境不再是“数据坟墓”，而是成为闭环中的一个节点。构建数据闭环的目的是让数据自动循环下去，为数据注入动力，避免数据“沉积”下来埋入坟墓，一个完善的数据闭环具有表 2-1 中的特征。

表 2-1 数据闭环的基本特征

	说 明	技术方案
松耦合	数据闭环中各个环节之间是松耦合的，彼此之间互不影响	设立数据缓冲区
自动化	数据的循环流动由系统自动完成，将人工作业降到最低	定制化数据对接系统、ETL 作业和调度工具
易扩展	数据闭环需要有良好的扩展性，当新的应用系统需要将数据对接时，通过配置文件的方式即可实现	定制化数据对接系统
可监控	数据闭环的运行情况能够实时监控，并可通过短信、邮件进行预警	日志系统、使用 BI 进行可视化监控

数据闭环成功的关键在于松耦合、易扩展，设立数据缓冲区可以以极低的成本达成这一目标。所有需要数据交互的系统，都要先将数据存储和数据缓冲区中，然后从数据缓冲区中选择需要的数据进行加载，这既避免了多个系统之间的直接耦合，同时也具备了易扩展的特性，新的系统只需要按照数据缓冲区的格式要求将数据存储和数据缓冲区中即可。

通过定制化的数据对接系统，实现数据的自动识别、加载，并结合周期性的 ETL 作业和调度工具，可以实现数据缓冲区数据的自动出入，是数据闭环中的数据“自动”流转。

此外，通过 BI 工具和日志收集监控工具，可实现整个数据闭环的可视化监控，并可以

通过短信、邮件进行预警，这为数据闭环的持续健康运行提供了保障。

下面将从数据缓冲区、ETL 作业、监控预警等方面进行介绍。

## 2.2 数据缓冲区

数据缓冲区是处于生产环境和分析环境之间的中间区域，它是数据闭环中各个系统间的数据中转站，从各个系统接收原始数据，并将其暂存在对应的目录中。其他系统可以从数据缓冲区中获取需要的数据文件。

为了便于管理和迁移数据，我们规定存入数据缓冲区中的数据使用文本文件的格式，这样一来，数据缓冲区就可以使用一台或几台文件服务器实现。几乎所有的应用系统都支持文本文件的数据交互，新的系统可以轻松加入数据闭环之中。

数据缓冲区的一端连接生产环境中的大量应用系统，另一端连接分析环境中的数据平台，避免了生产环境和分析环境的相互影响，同时也为系统之间数据文件的交互制定了统一标准（见图 2-2）。

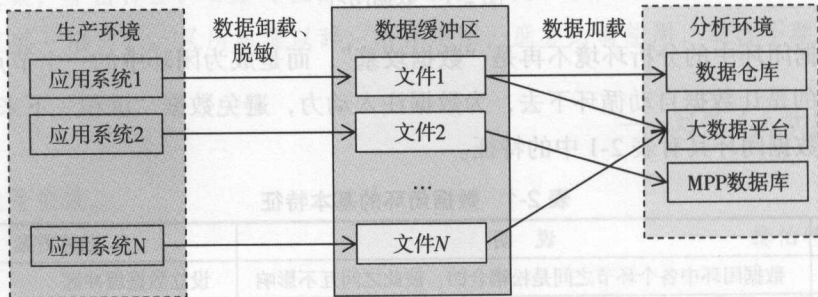


图 2-2 数据缓冲区连接生产环境和分析环境

数据缓冲区的另一个优势在于方便自动化和数据管理，多个应用系统的文件存档在同一个文件服务器中，便于数据的统一管理和分发。比如，在一个多部门、跨地域的企业中，不同地区、不同部门的数据文件之间的交互，如果没有数据缓冲区的统一收集与分发，那将会形成一个复杂的交叉网络。

表 2-2 列举了数据缓冲区的一些主要优点，本书主要专注于系统解耦，并基于数据缓冲区完成原始数据的自动加载过程。

表 2-2 数据缓冲区的优点

作用	备注
系统解耦	系统间互不影响
便于数据平台切换	基于文件，文件两端的数据平台互不影响

(续)

作用	备注
便于数据系统扩展	基于文件系统, 便于不同数据平台之间对接
便于数据管理	文件备份, 存储空间管理
便于数据分发	企业不同机构间的数据分发

## 2.2.1 系统解耦

### 1. 数据直连交互

数据直连的方式是将原始数据从生产系统直接导出至分析系统, 是数据交互的紧耦合方式。这种方式在数据规模较小时并不会出现问题, 因此很多企业搭建数据体系时采用了数据直连的方式, 但随着数据规模的增大, 数据直连的弊端会逐步展现, 让我们看一下如下的场景:

某个星期一的早晨, 数据工程师小王走入办公室, 发现开发工程师小李、系统工程师小周、开发经理和产品经理正聚在一起商讨问题。

开发工程师小李: 我昨晚上线完成后, 业务人员验证通过, 当时系统没有任何问题的, 程序肯定没问题!

产品经理: 可是现在系统反应奇慢无比, 基本处于瘫痪状态, 很多业务人员都等着开工呢, 怎么办?

系统工程师小周: 你自己看, 数据库服务器磁盘 I/O 好大……这种情况之前可是没有的……这个数据库进程是怎么回事? 磁盘 I/O 就是被它拖垮的。

开发经理 (一脸黑线): 赶紧查一查, 看看谁干的!

系统工程师小周: 这好像是分析数据库在抽数据。……小王你刚好来了, 你看这个作业是你的吧?

数据工程师小王 (紧张中): 这个我看看……平时都是 20 分钟就抽取结束了啊! 今天怎么还没有完成? 怎么回事?

开发经理: 那先赶紧停下来, 解决了生产问题再说。小王, 最近数据这块怎么老出问题啊!

数据工程师小王 (委屈): ETL 作业跑了半年多了, 都没问题。开发昨晚上线, 今天就出问题了……

开发工程师小李 (打断小王): 上线前都经过测试了, 上线后也验证了, 没有问题的。现在是数据库的问题, 和系统没有关系好吧?

数据工程师小王: 那为什么上线后, ETL 作业就这么慢呢?! 也不能怪我啊……



上面的场景是数据直连方式经常会遇到的问题，这种问题可能在生产系统上线后突然出现，也可能在平常的日子里莫名奇妙地发生。由于生产系统和分析系统之间的紧耦合，一旦出现问题，生产系统和分析系统都可能受到影响，而问题产生的原因却很难查清。表 2-3 总结了数据直连的弊端。

表 2-3 数据直连的弊端

弊端	说明
系统紧耦合	双方互相影响，生产系统切换影响分析系统，分析系统也会影响生产系统效率
不利于数据库权限管理	生产数据库需要为 ETL 作业开放权限，不利于生产数据的安全
不利于数据平台扩展	由于采用数据直连，其他数据平台（如 Hadoop 平台）加入时比较困难

2. 数据缓冲区交互

前面已经论述使用数据缓冲区进行交互的优点，本节将进一步研究数据缓冲区进行数据交互的详细流程，图 2-3 是整个过程的示意图。

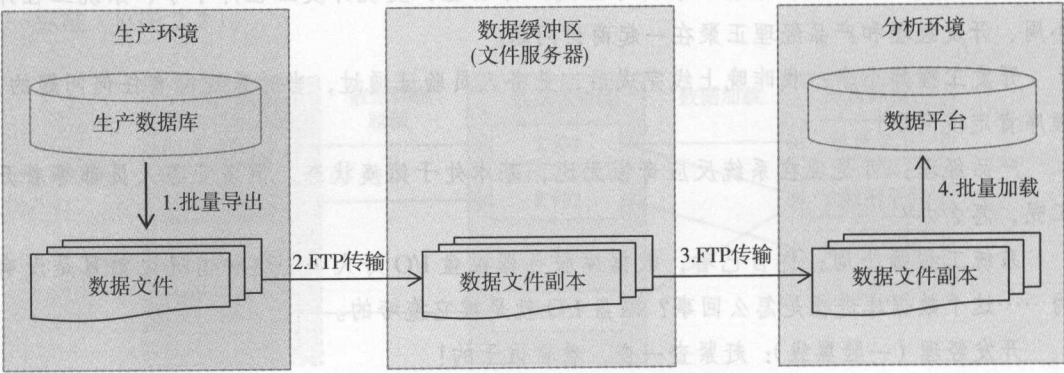


图 2-3 数据缓冲区进行数据交互的流程

在这个过程中，数据由生产环境流入分析环境，共经过以下四个步骤。

1) 批量导出。数据从生产数据库批量导出为文本文件，该过程使用 DBMS 系统自带的批量导出命令，对于大数据平台数据库，使用对应的命令或者使用第三方插件。2.2.2 节将对批量导出命令进行详细介绍。

2) FTP 传输第一阶段。将步骤 1) 导出的数据文件通过 FTP 上传至数据缓冲区。步骤 1) 和步骤 2) 的自动化过程可以通过 ETL 定时作业完成，实现方法参阅 ETL 作业章节。

3) FTP 传输第二阶段。将数据文件从数据缓冲区下载至分析环境中。

4) 批量加载。使用批量导入命令将文件加载至数据平台。

这种方式解决了职责不清的问题。从图 2-3 中可以发现，步骤 1) 和步骤 2) 处于数据缓冲区之前，它们属于生产环境的范畴，由开发工程师负责；步骤 3) 和步骤 4) 属于分析系统

的范畴，由数据工程师负责。

这个框架还解决了数据直连方式面临的以下问题。

1) 生产系统与分析系统的耦合问题。通过数据缓冲区实现了生产系统和分析系统的解耦，无论生产系统如何变更，只要传输至数据缓冲区中的文本文件格式不变，分析系统就不受影响；而分析系统在将数据加载至数据平台的时候，也不会影响到生产系统的性能。

2) 数据权限的问题，让数据更加安全。数据缓冲区的左边完全由开发工程师负责，因此生产数据库权限不会流转至后端的数据工程师；而通过在数据批量导出过程中对敏感数据的屏蔽处理（如对手机号码加密等）后，后端数据平台无法看到敏感数据，提高了数据安全性。

3) 增强了数据平台的可扩展性。由于各种数据平台，如传统数据仓库、Hadoop 平台、MPP 数据库等均对文本文件有良好的支持，不同平台之间的数据交互，均可以通过数据缓冲区实现数据交互。例如，Hadoop 平台的 Hive 数据仓库可以通过数据缓冲区与传统数据仓库中的关系数据库实现交互。

## 2.2.2 批量导出

批量导出是将数据库中的数据一次性导出至文本文件中，导出的文件有固定的列分隔符和行分隔符，或者有固定的字段长度。批量导出的方法大致可以分为以下两种（见表 2-4）。

1) 使用 ODBC 或 JDBC 接口，如 ETL 工具或定制的 Java 程序等。

2) 使用批量导出命令，一般是数据库自带的命令。

表 2-4 常用数据库批量导出方法

方法	举例	性能	适用场景
ODBC/JDBC 接口	ETL 工具、Java 程序	速度慢，对源数据库影响大	数据量较小的场景
批量导出命令	批量导出命令、hadoop shell 命令	速度快，对源数据库影响小	数据量大或跨系统的场景，如生产数据到分析数据，或数据仓库到大数据平台

表 2-5 中列出了常用数据库的批量导出命令，这些命令将在第 3 章详细介绍。

表 2-5 常用数据库批量导出命令

数据库系统	批量导出命令
SQL Server	bcp out
DB2	export
Oracle	第三方插件 sqlldr
MySQL	mysqldump
Hive	hadoop shell

1. SQL Server : bcp out

bcp 是 SQL Server 自带的批量导出 / 导入命令，它包括批量导出命令 bcp out 和批量导入命令 bcp in。bcp out 命令的语法如代码清单 2-1 所示（表 2-6 为参数说明）。

代码清单 2-1

```
bcp table_name out data_file
[-f format_file]
[-U login_id]
[-P password]
[-S [server_name[instance_name]]]
```

表 2-6 参数说明

参 数	参数含义
table_name	表名称
data_file	输出的数据文件的完整路径
-f format_file	指定格式化文件的完整路径
-U login_id	指定用于连接到 SQL Server 的登录 ID
-P password	指定登录 ID 的密码
-S[server_name[instance_name]]	指定要连接的 SQL Server 实例

例如，从 SQL Server 数据库表“dbo. 巡检商户明细”中批量导出数据到文件“巡检商户明细.dat”中的 bcp out 命令如代码清单 2-2 所示。

代码清单 2-2

```
bcp dbo. 巡检商户明细 out d:\ 巡检商户明细 .dat -fd:\ 巡检商户明细 .fmt -U test -P 123 -S localhost
```

\* 注：实际输入命令时，应在一行，中间不能有换行。

代码清单 2-1 中各参数说明如下。

- 1) dbo. 巡检商户明细是需要导出数据的表名称。
- 2) d:\ 巡检商户明细 .dat 为导出的文件路径及文件名称。
- 3) -f d:\ 巡检商户明细 .fmt 指明格式文件的路径及名称。
- 4) -U test 指明登录数据库所用的用户名称为“test”。
- 5) -P 123 指明数据库用户“test”的密码为“123”。
- 6) -S localhost 指明登录的数据库为“localhost”。

bcp out 命令使用了格式文件选项“-f format\_file”，用来指明数据文件的格式。格式文件是用来描述数据文件格式的文件，在格式文件中需指明要导出的字段名称、长度、列分割符、行分隔符、排序方式等，可以用在 bcp out 和 bcp in 命令中。用在 bcp out 命令中时，

它用来定义导出文件的格式；用在 bcp in 命令中时，它用来描述待导入文本文件的格式。

图 2-4 所示为格式文件的式样。

Version	Host file data type	Host file data length	Server column order	Column collation
9.0				
Number of columns				
4				
Host file field order				
1	SQLCHAR	0	1	DepartmentID
2	SQLCHAR	0	2	Name
3	SQLCHAR	0	3	GroupName
4	SQLCHAR	0	4	ModifiedDate
	Prefix length	Terminator	Server column name	
	0	"\t"		
				SQL_Latin1_General_CP1_CI_AS
				SQL_Latin1_General_CP1_CI_AS
				"

图 2-4 SQL Server 控制文件格式

对图 2-4 中涉及参数的说明如下：

- 1) Version 是微软公司对 SQL Server 系列产品的版本编号，如表 2-7 所示。
- 2) Number of columns 是所要导出的数据库表所含的字段个数。
- 3) Host file field order 是输出的数据文件字段列的序号，与 Server column order 对应，一般与 Server column order 保持一致。
- 4) Host file data type 保持默认值“SQLCHAR”即可。
- 5) Prefix length（前缀长度），一般导出的数据文件字段都不填充前缀，因此此处为 0。
- 6) Host file data length 为输出文件字段的长度，此处如果 Terminator 指明的分隔符不为“”，则该参数并不起实际作用。
- 7) Terminator 指明数据文件的列分隔符和行分隔符（格式文件最后一行该字段为行分隔符）。
- 8) Server column order 为数据表中字段的数据，与 Host file field order 保持一致。
- 9) Server column name 是数据库中的字段名称。
- 10) Column collation 指明字段的 collation，一般仅为字符格式的字段使用，保持默认即可。

表 2-7 SQL Server 版本号对照表

产品名称	版本号
SQL Server 2014	12
SQL Server 2012	11
SQL Server 2008 R2	10
SQL Server 2008	10
SQL Server 2005	9
SQL Server 2000	8



格式文件中应重点关注的部分为 Number of columns、Host file field order、Terminator、Server column order、Server column name，其余保持默认即可。

如何制作格式文件呢？下面使用一个具体的例子说明如何创建格式文件。假设在数据库中创建表“dbo.巡检商户明细”，其创建的表脚本如代码清单 2-3 所示。

代码清单 2-3

```
CREATE TABLE [dbo].[ 巡检商户明细 ]
(
    [ 商户代码 ]      [varchar] (30)      NULL,
    [ 商户名称 ]      [nvarchar] (100)    NULL,
    [ 机构号 ]        [varchar] (20)      NULL,
    [ 合作方名称 ]    [nvarchar] (100)    NULL,
    [ 分公司 ]        [nvarchar] (50)     NULL
);
```

使用 bcp format 命令，可得到该表的控制文件模板“巡检商户明细 .fmt”，命令如代码清单 2-4 所示。

代码清单 2-4

```
bcp dbo. 巡检商户明细 format nul -c -f 巡检商户明细 .fmt -Utest -P123 -Slocalhost
```

巡检商户明细 .fmt 文件中默认的列分隔符为“\t”（制表符）、行分隔符为“\r\n”（回车换行符），其内容如代码清单 2-5 所示。

代码清单 2-5

```
9.0
5
1  SQLCHAR 0      30      "\t"      1      [ 商户代码 ]
   SQL_Latin1_General_CP1_CI_AS
2  SQLCHAR 0      100     "\t"      2      [ 商户名称 ]
   SQL_Latin1_General_CP1_CI_AS
3  SQLCHAR 0      20      "\t"      3      [ 机构号 ]
   SQL_Latin1_General_CP1_CI_AS
4  SQLCHAR 0      100     "\t"      4      [ 合作方名称 ]
   SQL_Latin1_General_CP1_CI_AS
5  SQLCHAR 0      50      "\r\n"    5      [ 分公司 ]
   SQL_Latin1_General_CP1_CI_AS
```

注：格式文件的最后一行必须为空白行，否则在使用 bcp 命令时会报格式错误。

得到格式文件模板之后，就可以在此基础上进行修改了，比如使用“#”作为列分隔符，使用“\n”作为行分隔符，修改后的格式文件内容如代码清单 2-6 所示。

3. Oracle : sqlldr2 代码清单 2-6

```

9.0
5
1  SQLCHAR  0  30  "\""  1  [ 商户代码 ]  SQL_Latin1_General_CP1_CI_AS
2  SQLCHAR  0  100  "\""  2  [ 商户名称 ]  SQL_Latin1_General_CP1_CI_AS
3  SQLCHAR  0  20  "\""  3  [ 机构号 ]  SQL_Latin1_General_CP1_CI_AS
4  SQLCHAR  0  100  "\""  4  [ 合作方名称 ]  SQL_Latin1_General_CP1_CI_AS
5  SQLCHAR  0  50  "\"n"  5  [ 分公司 ]  SQL_Latin1_General_CP1_CI_AS

```

这样,将修改后的格式文件用于 bcp out 命令,则输出的数据文件将使用“#”作为列分隔符,使用“\n”作为行分隔符。作为验证,可以执行如代码清单 2-7 所示的 bcp out 命令。

代码清单 2-7

```

bcp dbo. 巡检商户明细 out D:\ 巡检商户明细 .dat -f D:\ 巡检商户明细 .fmt -Utest -P123
-Slocalhost

```

查看输出的数据文件“巡检商户明细数据.dat”,其内容如代码清单 2-8 所示。

代码清单 2-8

```

000391952# 新白鹿餐厅 (百联中环店) #0880# 安智餐饮有限公司 # 安徽分公司
000472873# 颐和四季体验馆 #0880# 万源城娱乐城 # 北京分公司
000901032# 书院人家 #0880# 万家灯火餐饮文化传播公司 # 北京分公司
000900109# 三阳湾食府 #0880# 三阳众城文化管理有限公司 # 北京分公司
.....

```

从以上代码清单中可以看到,输出的文本文件使用“#”作为列分隔符,使用“\n”作为行分隔符。可以根据需要修改格式文件,从而得到满足要求的数据文件输出。更详细的 bcp 命令可参阅微软官方帮助文档。

## 2. DB2 : export

IBM DB2 数据库中数据的批量导出可以使用 export 命令,其基础语法如代码清单 2-9 所示(表 2-8 为参数说明)。

代码清单 2-9

```

export to filename of {ixf |del | wsf }
[ modified by {filetype-mod ...} ] { select-statement |[ where ... ]}

```

例如,在 DB2 中自带 sample 数据库中的一张表,其创建的表脚本如代码清单 2-10 所示。

表 2-8 DB2 export 命令的参数说明

参 数	参数含义
filename	导出的文件名称
IXF DEL WSF	输出格式，对于文本文件，选择 DEL 格式
filetype-mod	文件类型修饰符，包括设置字符串界定符、列分隔符等多种选项
chardel	指定字符串界定符
coldel	指定列分隔符
select-statement	用于提取数据的 select 语句

代码清单 2-10

```
CREATE TABLE HB_STATIC (  
    STA_MTH          VARCHAR(8)          DEFAULT NULL,  
    FACE_VALUE       INTEGER             DEFAULT 5,  
    TOTAL_AMT        BIGINT              DEFAULT 0  
);
```

现在需要把这张表中的数据导出成文本文件，运行 db2cmd，依次运行如代码清单 2-11 所示的命令。

代码清单 2-11

```
db2 connect to sample  
db2 export to d:\\hb_static.del of del modified by chardel' ' coldel; select *  
from hb_static
```

export 命令指明导出文件路径及名称“d:\\hb\_static.del”，chardel “指明使用”作为字符串界定符（数据库表中的字符类型的数据使用该字符串包裹），coldel; 指明使用；作为列分隔符。最终得到的文本文件 d:\\hb\_static.del 的内容（部分）如代码清单 2-12 所示。

代码清单 2-12

```
'201402';20;480  
'201402';5;165  
'201402';10;200  
'201402';50;1100  
'201403';5;915
```

由于 DB2 的 export 命令没有提供形如 SQL Server 格式文件之类的控制文件，而仅通过命令选项指定列分隔符、字符串界定符，使得 DB2 的 export 命令导出的文本文件格式较为单一（例如，只能使用一个字符作为列分隔符），但在大多数场景中，export 可以满足要求。

### 3. Oracle : sqlldr2

Oracle 数据库未提供批量导出命令，一般采用第三方工具进行批量导出。SQLPLUS 提供的 Spool 工具虽然可以进行数据导出，但是它并不适合大量数据的快速导出，主要原因是其导出效率很低，大量数据导出会非常耗时。

另一款批量导出工具 sqlldr2 适用于大批量数据的导出，速度非常快，可以将数据以 csv、txt 等格式导出。

首先需要下载 sqlldr2.exe（可上网搜索），如果安装的是 64 位的 Oracle，则需要下载 sqlldr264.exe，然后将 sqlldr2.exe 复制到 \$ORACLE\_HOME 的 BIN 目录（该目录中有 Oracle 自带的 sqlldr.exe，这是 Oracle 的批量导入工具。没错，Oracle 提供了批量导入工具。却没有提供批量导出工具）中。现在就可以开始使用 sqlldr2.exe 了，sqlldr2 的命令格式如代码清单 2-13 所示（表 2-9 为其参数说明）。

代码清单 2-13

```
sqlldr2 logon_str {query="select_statement" | sql=sql_file }
[file=output_file]
[field=col_del]
[record=row_del]
[quote=quote]
```

表 2-9 sqlldr2 的参数说明

参 数	含 义
logon_str	数据库登录信息，必需参数
select_statement	查询语句，用于提取数据，与 sql_file 两者二选一
sql_file	指定的 sql 语句脚本文件
output_file	输出的数据文件路径及名称，如果不指定此选项，则默认输出 uldrdata.txt
col_del	列分隔符，默认为逗号
row_del	行分隔符，默认为 \r\n
quote	字符串界定符

例如，在 Oracle 数据库中有一张表，其创建表的脚本如代码清单 2-14 所示。

代码清单 2-14

```
CREATE TABLE "ODS"."DP_COMMENT"
(
  "MEMBERID"          VARCHAR2(50),
  "TASTE"              VARCHAR2(50),
  "ENVIRONMENT"        VARCHAR2(50),
  "SERVICE"           VARCHAR2(50),
  "LEVEL_SCORE"        VARCHAR2(50),
  "CONTENT"            VARCHAR2(2000),
```



```
"SHOPID"          VARCHAR2(50)
);
```

使用 sqlldr2 将表中的数据导出至文本文件 DP\_COMMENT.txt 中，如代码清单 2-15 所示。

代码清单 2-15

```
sqlldr2 ods/ods@yfb_orc query="select * from ODS.DP_COMMENT"
file=d:\\DP_COMMENT.txt field=##$
```

sqlldr2 命令将表中的数据导出至 DP\_COMMENT.txt，field=##\$ 指明导出的文本文件中的列使用 ##\$ 进行分割。查看文件 DP\_COMMENT.txt，其内容如代码清单 2-16 所示(部分)。

代码清单 2-16

```
195084790##$##$##$##$##$40##$ 菜品味道中规中矩，价钱稍贵！ ##$17222108
630568##$4##$4##$4##$50##$ 很满意 这是很好的一次体验 谢谢 ##$17222108
178216498##$3##$3##$3##$50##$ 尝试了真不错 ##$17222108
```

该命令的一个很有用的选项为 table 选项，该选项可以生成一个默认的控制文件，该控制文件可以用于 Oracle 的 sqlldr 命令进行数据批量导入。在上述导出的命令中加入 table 选项，如代码清单 2-17 所示。

代码清单 2-17

```
sqlldr2 ods/ods@yfb_orc query="select * from ods.dp_comment"
file=d:\\dianping_comment.txt field=##$ table=dp_comment
```

执行上述命令后，除了输出文件 d:\\DP\_COMMENT.txt 外，还生成了控制文件 dp\_comment\_sqlldr.ctl，代码清单 2-18 是 dp\_comment\_sqlldr.ctl 的内容(略做了修改，删除了注释部分)。

代码清单 2-18

```
load data
infile 'd:\\dp_comment.txt'
insert into table dp_comment
fields terminated by x'2324' trailing nullcols
(
  "memberid"      char(50)          nullif "memberid"=blanks,
  "taste"          char(50)          nullif "taste"=blanks,
  "environment"    char(50)          nullif "environment"=blanks,
  "service"        char(50)          nullif "service"=blanks,
```

```

"level_score" char(50) nullif "level_score"=blanks,
"content" char(2000) nullif "content"=blanks,
"shopid" char(50) nullif "shopid"=blanks
)

```

代码清单 2-18 中, x'2324' 是十六进制的字符 #\$(由上文中 sqlldr2 的 field 选项指明)。上述控制文件可以用于数据批量导入 Oracle 数据库中, 控制文件用于批量导入的方法请参阅批量导入章节的内容。

#### 4. Hive : hadoop fs

Hive 是 Hadoop 平台上一款非常流行的数据仓库分析工具, 由于类似 SQL 的语言风格, 使得其学习成本很低, 所以是大数据分析的必学工具。

Hive 数据导出, 主要的应用场景是将 Hive 表中的数据导出到 Linux 操作系统中, 然后供其他数据产品使用。该导出过程可以使用 ETL 工具(参阅第 2.3 节), 也可以使用 hadoop shell 命令完成。

例如, 现在有一张 Hive 表, 其创建表的脚本如代码清单 2-19 所示(未列出全部字段)。

代码清单 2-19

```

create table adobe_nwd_prd
(
    accept_language string comment '浏览器中可接受的语言标题',
    browser bigint comment '实际用于单击的浏览器 ID',
    domain string comment '用户 ISP 域',
    duplicate_events string comment '列出计为重复的每个事件',
    .....
)
comment 'adobe pc 端原始数据'
partitioned by(load_day string)
row format delimited
fields terminated by '\t'
stored as textfile;

```

Hive 表使用了 load\_day 字段作为 partition 字段, 即每天的数据存放在一个 partition 中, 通过 hive shell 命令可以查看当前表中的 partition 情况, 如代码清单 2-20 所示。

代码清单 2-20

```

hive> show partitions adobe_nwd_app;
OK
load_day=20150908
load_day=20150909
load_day=20150910
load_day=20150911
load_day=20150912

```

```
load_day=20150913
load_day=20150914
Time taken: 0.379 seconds, Fetched: 7 row(s)
```

每个 partition 对应一个 hdfs 目录，按照 1.2.3 节中“Hive 分区表与增量更新”中的规则，Hive 表的数据存放路径如代码清单 2-21 所示。

代码清单 2-21

```
[root@qzy ~]# hadoop fs -ls /data/adobe.ADOBE_NWD_APP
Found 7 items
drwxr-xr-x -root supergroup 0 2015-09-09 00:40
/data/adobe.ADOBE_NWD_APP/20150908
drwxr-xr-x - root supergroup 0 2015-09-10 00:39
/data/adobe.ADOBE_NWD_APP/20150909
drwxr-xr-x - root supergroup 0 2015-09-11 06:18
/data/adobe.ADOBE_NWD_APP/20150910
drwxr-xr-x - root supergroup 0 2015-09-12 00:33
/data/adobe.ADOBE_NWD_APP/20150911
drwxr-xr-x - root supergroup 0 2015-09-13 00:33
/data/adobe.ADOBE_NWD_APP/20150912
drwxr-xr-x - root supergroup 0 2015-09-14 00:38
/data/adobe.ADOBE_NWD_APP/20150913
drwxr-xr-x - root supergroup 0 2015-09-14 10:33
/data/adobe.ADOBE_NWD_APP/20150914
```

使用代码清单 2-22 所示命令将 partition (load\_day=20150908) 的所有数据导出至 Linux 系统的本地目录 /tmp/datafiles 中。

代码清单 2-22

```
hadoop fs -copyToLocal /data/adobe.ADOBE_NWD_APP/20150908 /tmp/datafiles
```

通过程序可以循环导出指定时间范围内的数据，第 3 章将提供一种 Java 批量导出 Hive 数据的多线程实现。

上述方式实现的是不含条件参数的批量导出，如果希望导出 where 条件限定的数据，则需要将数据事先生成一张中间表，然后将此中间表的全部数据导出，此处不再赘述。

## 2.2.3 FTP 传输

由于数据缓冲区实际上是文件服务器，在内网环境中使用 FTP 进行传输是一种很方便的方式。在数据闭环中，FTP 传输连接了数据缓冲区的上游和下游，稳定高效的文件传输对整个数据闭环起重要作用。

对数据工程师来说，FTP 自动传输可以通过 ETL 工具、命令行、定制程序等方式实现。

ETL 工具一般都自带文件传输模块，可以直接使用 FTP 文件传输功能。例如，开源 ETL 工具 Pentaho Kettle 的作业功能中，即有文件传输模块，包含 FTP 上传、FTP 下载等组件，图 2-5 所示的为 Kettle 的文件传输组件。

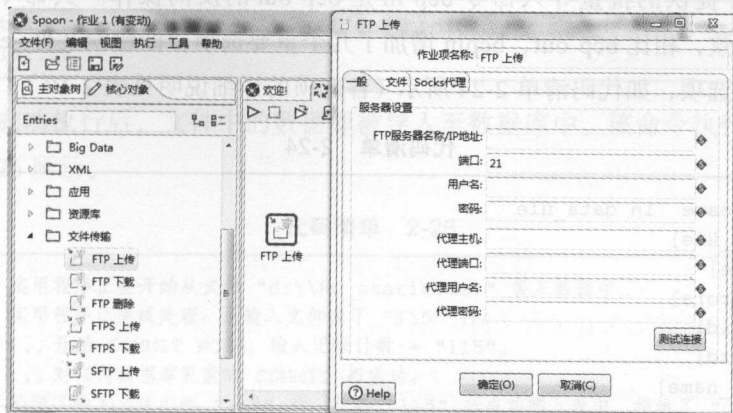


图 2-5 Kettle 中的文件传输模块

但 ETL 工具提供的 FTP 传输模块有一个局限，就是待传输的文件名如果是动态的，例如文件名称以日期作为后缀，或者根据条件选择不同的文件进行传输，则使用 ETL 工具实现起来会比较困难（虽然 ETL 工具本身也提供简单的参数输入及文件名称表达式匹配，但总体实现成本比其他两种方式要高很多）。

一种替代的方案是使用脚本语言，也就是通过脚本将 FTP 命令包装起来，以实现参数的传入，解决动态文件名问题。例如在 Linux 系统上，使用 shell 脚本或 Python 脚本，调用 FTP 命令来实现文件的 FTP 上传或 FTP 下载。

本书推荐的方式是使用高级程序语言进行 FTP 文件传输的包装，因为高级程序语言一般都提供 FTP 文件传输的程序包，可以根据需要记录传输日志，并且可以方便实现多线程 FTP 文件传输，提高传输效率。比如 Java 的 commons-net-x.x.x.jar (x.x.x 代表版本号) 包就提供了 FTP 命令的 API 接口，代码清单 2-23 是该接口的调用示例（具体实现请参阅第 3 章）。

代码清单 2-23

```
import org.apache.commons.net.ftp.FTPClient;
FTPClient ftpClient = new FTPClient();
ftpClient.connect(serverIP);
ftpClient.login(user, passsWord);
ftpClient.retrieveFile(remoteFileName, new FileOutputStream(localFilePath));
```



## 2.2.4 批量导入

### 1. SQL Server : bcp in

SQL Server 提供的批量导入命令 `bcp in` 是 `bcp out` 的反向操作。其命令的格式与 `bcp out` 的也基本一致，相比 `bcp out`，`bcp in` 增加了几个重要选项，即错误文件 `-e` 选项、最大允许错误行数 `-m` 选项，如代码清单 2-24 所示（各选项的详细说明见表 2-10）。

代码清单 2-24

```
bcp table_name in data_file
[-f format_file]
[-e err_file]
[-m max_errors]
[-U login_id]
[-P password]
[-S server_name]
```

表 2-10 bcp in 的选项说明

选 项	说 明
-e err_file	用于指定错误文件的完整路径，此文件用于存储 bcp 实用工具无法从文件传输到数据库的所有行，相当于错误日志
-m max_errors	用于指定 bcp 命令允许出现的最大错误行数，错误行数未达到 max_errors 时，bcp 导入将继续进行，默认值为 10
-f format_file	用于指定格式化文件的完整路径，格式化文件用于指定行列分隔符、是否跳过某些列等

`-m` 选项在批量导入大文件时是非常有用的，由于一些数据会含有少量的错误数据，而这些错误数据并不影响整体的数据效果，这些场景本身对数据的完整性要求并不高（不同于交易明细数据），这时使用 `-m` 选项指定一个阈值，当错误行数小于阈值的时候，`bcp` 继续执行，可以将正确的数据导入，确保业务的正常进行。

`-m` 选项通常与 `-e` 选项配合使用，可以从 `-e` 选项指定的 `err_file` 中查看出现错误的数行。

目前许多互联网公司网站中会嵌入网站日志分析工具，实时收集用户的单击行为，如 Adobe omniture、WebTrends 等，这些工具会产生大量日志数据，而这些日志数据中不可避免地存在部分无法正常导入数据库的记录，通过 `-m` 选项可以跳过这些错误数据，从而保证绝大部分数据可用。

### 2. DB2 : import

DB2 批量数据导入可使用 `db2 import` 命令，该命令是 `db2 export` 命令的反向命令，其格式与 `db2 export` 命令的一致。例如将第 2.2.2 节中“DB2：export”导出的文本文件“d:\

hb\_static.del”再导入至表 HB\_STATIC2 (该表结构与表 HB\_STATIC 的相同)中,方法如代码清单 2-25 所示。

代码清单 2-25

```
db2 import from d:\\hb_static.del of del modified by chardel'' coldel; insert
into HB_STATIC2
```

上述命令成功执行后,文件中的数据即被导入至数据库中。该命令执行后的反馈信息如代码清单 2-26 所示。

代码清单 2-26

```
SQL3109N 实用程序正在开始从文件 "d:\\hb_static.del" 装入数据中。
SQL3110N 实用程序已完成处理。从输入文件读了 "115" 行。
SQL3221W ... 开始 COMMIT WORK。输入记录计数 = "115"。
SQL3222W ... 对任何数据库更改的 COMMIT 都成功。
SQL3149N 处理了输入文件中的 "115" 行。已将 "115" 行成功插入表中。拒绝了 "0" 行。
读取行数          = 115
跳过行数          = 0
插入行数          = 115
更新行数          = 0
拒绝行数          = 0
落实行数          = 115
```

### 3. Oracle : sqldr

Oracle 自带的批量导入工具 sqldr 可以实现数据的快速批量导入,其命令格式如代码清单 2-27 所示(表 2-11 为其参数说明)。

代码清单 2-27

```
sqldr logon_str control=ctr_file log=log_file bad=bad_file errors=max_errors
```

表 2-11 sqldr 的参数说明

参 数	说 明
logon_str	数据库登录信息,必需参数
ctr_file	控制文件完整路径
log_file	日志文件完整路径
bad_file	错误文件完整路径
max_errors	最大允许错误行数

其中控制文件的作用类似于 SQL Server 中的格式文件,可以通过此控制文件指明输入文本的格式,以及导入数据库时的加载方式,图 2-6 所示的为 Oracle 控制文件的格式说明。

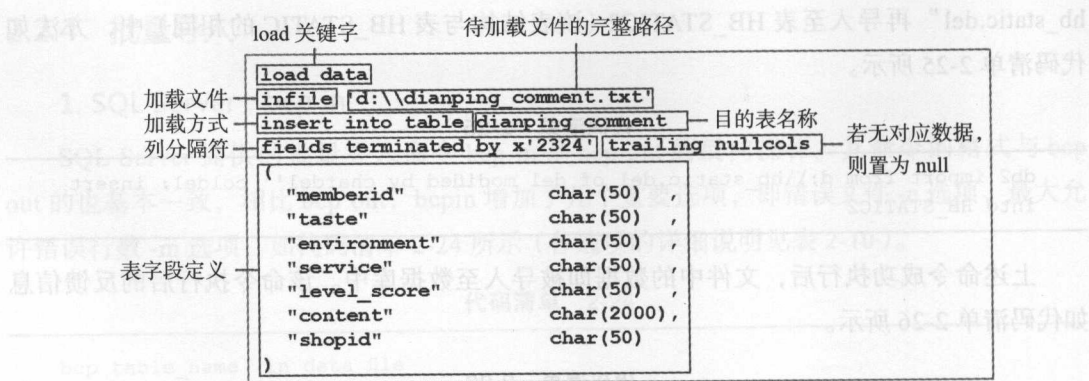


图 2-6 Oracle 控制文件的格式

Oracle sqlldr 批量导入有多种方式，表 2-12 对此做了总结。

表 2-12 sqlldr 的四种导入方式

导入方式	说明
append into	若原表有数据，则在后面追加数据
insert into	装载空表，如果原表有数据，则 sqlldr 会停止加载，此为默认值
replace into	原表数据全部删除后加载
truncate into	功能和 replace 的相同，会用 truncate 语句删除原有数据

下面尝试将 Adobe omniture 对某网站的监控日志导入 Oracle 的数据库中，命令如代码清单 2-28 所示。

代码清单 2-28

```
sqlldr ods/ods@yfb_orc control=e:\Adobe\adobe_src_data.ctl
log=e:\Adobe\log.txt bad=e:\Adobe\error_record.txt errors=1000000
```

由于该文件是对网站单击行为的日志记录，因此对错误行的容忍度是比较大的，为了保证能够将正确的数据导入，我们设置 `errors=1000000`，即允许导入过程出现 1000000 条错误记录，这些错误记录会被记录到 `bad` 选项指定的“`e:\Adobe\error_record.txt`”中。

控制文件“`e:\Adobe\adobe_src_data.ctl`”指明了数据文件、文件分隔符信息、加载方式等内容，如代码清单 2-29 所示。

代码清单 2-29

```
load data infile 'E:\Adobe\01-niwodai-prd_2015-07-26.tsv'
append into table adobe_src_data
fields terminated by '\t'
trailing nullcols
(
  accept_language
```

```
,browser
,browser_height
.....
)
```

上述 sqldr 命令在 Windows 的 cmd 窗口中执行后, 查看 log 文件 “e:\Adobe\log.txt” 的内容, 如代码清单 2-30 (仅列出部分内容) 所示。

代码清单 2-30

```
SQL*Loader: Release 11.2.0.1.0 - Production on 星期五 7月 31 09:22:22 2015
Copyright (c) 1982, 2009, Oracle and/or its affiliates. All rights reserved.
控制文件: e:\Adobe\adobe_src_data.ctl
数据文件: e:\Adobe\01-niwodai-prd_2015-07-26.tsv
错误文件: e:\Adobe\error_record.txt
废弃文件: 未作指定
```

(可废弃所有记录)

要加载的数: ALL

要跳过的数: 0

允许的错误: 1000000

绑定数组: 64 行, 最大 256000 字节

继续: 未作指定

所用路径: 常规

表 ADOBE\_SRC\_DATA, 已加载从每个逻辑记录

插入选项对此表 APPEND 生效

TRAILING NULLCOLS 选项生效

列名	位置	长度	中止	包装数据类型
ACCEPT_LANGUAGE	FIRST	*	WHT	CHARACTER
BROWSER	NEXT	*	WHT	CHARACTER
.....				

记录 4245: 被拒绝 - 表 ADOBE\_SRC\_DATA 的列 USER\_AGENT 出现错误。

数据文件的字段超出最大长度

.....

表 ADOBE\_SRC\_DATA:

417707 行 加载成功。

由于数据错误, 所以 913 行 没有加载。

由于所有 WHEN 子句失败, 所以 0 行 没有加载。

由于所有字段都为空的, 所以 0 行 没有加载。

为绑定数组分配的空间: 168216 字节 (1 行)

读取缓冲区字节数: 1048576

跳过的逻辑记录总数: 0

读取的逻辑记录总数: 418620

拒绝的逻辑记录总数: 913

废弃的逻辑记录总数: 0

从 星期五 7月 31 09:22:22 2015 开始运行

在 星期五 7月 31 10:19:56 2015 处运行结束

经过时间为: 00: 57: 34.13

CPU 时间为: 00: 03: 21.71



从以上代码中可以看到，日志文件记录了很多重要信息，如 `sqlldr` 命令中指定的控制文件（`control=e:\Adobe\adobe_src_data.ctl`）、错误文件（`bad= e:\Adobe\ error_record.txt`）、允许的错误的条数（`errors=1000000`）。

日志还记录了控制文件指定的数据文件（`e:\Adobe\01-niwodai-prd_2015-07-26.tsv`）、插入选项对表 `APPEND` 生效（即加载方式，`append into`）、`TRAILING NULLCOLS` 选项生效（`trailing nullcols`，该选项生效时，当数据文件中出现连续两个列分隔符时，对应字段值将被置为 `null`）。

在罗列了表中所有字段信息之后，日志文件记录了被拒绝的记录在数据文件中所在的行，以及被拒绝的原因，之后的信息还展示了加载成功的行数，以及由于错误被拒绝的行数，最后记录了导入该文件的耗时为 57 分钟 34 秒，之所以消耗这么长时间，是由于数据文件每行包含了 265 个字段，且字段长度都比较大。

日志文件有很多潜在的用途，通过程序读取日志文件可以将其中的重要信息展示在页面上，或写入日志数据库，从而便于作业的管理和监控。对日志文件的处理是数据闭环监控中的重要手段之一。

#### 4. Hive : add partition

Hive 有多种批量加载方式，根据数据文件存放的位置不同，Hive 加载数据面临两种情形：从本地文件系统加载数据以及从 HDFS 中加载数据。Hive shell 提供 `load data` 命令可以完成上述两种情形下的数据批量导入，如代码清单 2-31 所示。

代码清单 2-31

---

```
load data [local] inpath 'data_file' into table table_name;
```

---

当导入本地文件至 Hive 表中时，需要指明 `local` 关键字，并且随后的 `data_file` 参数用于指明基于本地文件系统的完整文件路径；当导入 `hdfs` 文件至 Hive 表中时，不需要 `local` 关键字，且 `data_file` 为 `hdfs` 文件系统的文件路径或 `hdfs` 文件 `url`。

为了便于程序化实现，这里采用 Hadoop shell 结合 Hve shell 的方式实现 Hive 表的批量导入。

根据第 1 章的 Hive 表更新规则，我们通过两个步骤完成数据的批量导入。首先通过 Hive shell 为 Hive 表增加一个 `partition`，并指定 `location`；然后使用 Hadoop shell 将文件 `copy` 至该 `partition` 对应的 `location` 目录，图 2-7 展示了这个过程。

按照图 2-7 所示的方式，我们尝试将上述 Adobe 数据文件“`e:\Adobe\01-niwodai-prd_2015-07-26.tsv`”导入至 Hive 表中。

为了确保 `location` 指定的 `hdfs` 目录存在，先执行 `hadoop shell` 命令，创建一个目录，如代码清单 2-32 所示。

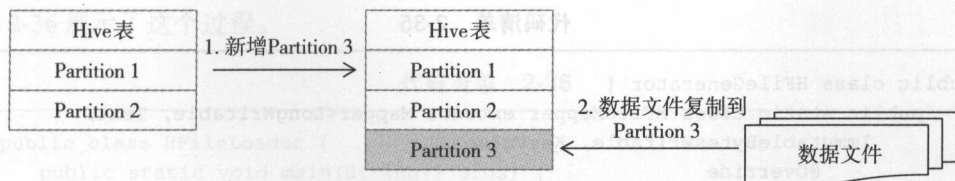


图 2-7 Hive shell + Hadoop shell 批量导入数据至 Hive 表

代码清单 2-32

---

```
hadoop fs -mkdir /data/adobe_log_app/20150726
```

---

下面就可以分两步完成数据的导入。首先为表 `adobe_log_app` 增加一个新的 `partition`，在 `hive shell` 环境中，执行如代码清单 2-33 所示的命令。

代码清单 2-33

---

```
hive>alter table adobe_log_app add partition(load_day='20150726') location
'/data/adobe_log_app/20150726'
```

---

然后，使用 `hadoop shell` 将文件复制到 `hdfs` 目录 “`/data/adobe_log_app/20150726`” 中，如代码清单 2-34 所示。

代码清单 2-34

---

```
hadoop fs -copyFromLocal
/usr/queziyang/data/01-niwodai-prd_2015-07-26.tsv
/data/adobe_log_app/20150726
```

---

至此，即完成了数据文件从本地文件系统批量导入至 `Hive` 表中，整个过程相对比较耗时的操作仅出现在数据文件复制的过程中，且复制效率要比 `hive shell` 的 `load data` 的高。上述的各个命令，可以通过编程语言调用，从而实现自动化。

## 5. Hbase : bulk load

`Hbase` 的数据加载方式也有很多种，但最高效的方式是使用 `Hbase` 的 `bulk load` 命令。`Hbase` 的 `bulk load` 命令分为两步：

- 1) 使用一个 `MapReduce` 作业将数据转换为 `Hbase` 的内部数据格式。
- 2) 将生成的 `StoreFiles` 直接加载到 `Hbase` 集群中。

代码清单 2-35 展示了 `MapReduce` 生成 `HFile` 的过程，这个过程仅包含 `Map`，不需要 `Reduce`。

## 代码清单 2-35

```

public class HFileGenerator {
    public static class HFileMapper extends Mapper<LongWritable, Text,
        ImmutableBytesWritable, KeyValue> {
        @Override
        protected void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
            // 获取 RowKey
            ImmutableBytesWritable rowkey = new
                ImmutableBytesWritable(items[0].getBytes());
            // 此处添加处理输入数据文件的代码……
            // 按照 org.apache.hadoop.hbase.KeyValue 的格式输出
            KeyValue kv = new KeyValue(Bytes.toBytes(items[0]),
                Bytes.toBytes("item"), Bytes.toBytes(column), System.currentTimeMillis(),
                Bytes.toBytes(prefValue));
            context.write(rowkey, kv);
        }
    }

    public static void main(String[] args) throws IOException,
        InterruptedException, ClassNotFoundException {
        Configuration conf = new Configuration();
        String[] dfsArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
        Job job = new Job(conf, "HFile bulk load Job");
        job.setJarByClass(HFileGenerator.class);
        job.setMapperClass(HFileMapper.class);
        job.setReducerClass(KeyValueSortReducer.class);
        job.setMapOutputKeyClass(ImmutableBytesWritable.class);
        job.setMapOutputValueClass(KeyValue.class);
        job.setPartitionerClass(SimpleTotalOrderPartitioner.class);
        FileInputFormat.addInputPath(job, new Path(dfsArgs[0]));
        FileOutputFormat.setOutputPath(job, new Path(dfsArgs[1]));
        HFileOutputFormat.configureIncrementalLoad(job, hbaseTableName);
        System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
}

```

代码主要包含 HFileMapper 和一个 main() 函数，在 HFileMapper 的 map 方法中处理输入数据文件，并按照 org.apache.hadoop.hbase.KeyValue 的格式输出。

在 main() 函数中，FileInputFormat.addInputPath 用于指定输入文件的路径（HDFS 路径），FileOutputFormat.setOutputPath 则用于指定 MapReduce 作业的输出路径，即生成的 HFile 最终的存放路径。

最后通过 HFileOutputFormat.configureIncrementalLoad(job, hbaseTableName) 导入 hbase 表的相关信息，从而使得 Map 的输出最终与 hbaseTableName 相匹配。

上述过程在指定的输出路径中生成 HFile 文件，可以通过 org.apache.hadoop.hbase.mapreduce.LoadIncrementalHFiles 的 doBulkLoad 方法将其挂载到对应的 hbase 表中，代码

清单 2-36 展示了这个过程。

代码清单 2-36

```
public class HFileLoader {
    public static void main(String[] args) {
        String[] dfsArgs = null;
        try {
            dfsArgs = new
                GenericOptionsParser(HbaseUtils.getConfiguration(),
                    args).getRemainingArgs();
            LoadIncrementalHFiles loader = new
                LoadIncrementalHFiles(HbaseUtils.getConfiguration());
            loader.doBulkLoad(new Path(dfsArgs[0]),
                HbaseUtils.getTable(hbaseTableName));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

这个过程非常简单，只需要指明已经生成的 HFile 的路径，其他的工作交给 doBulkLoad 方法即可。加载非常快，因为它实际上是执行类似 hadoop 的 mv 操作。

bulk load 主要的耗时阶段在于生成 HFile，而在加载阶段则非常迅速。这种方式比较适用于往 Hbase 空表中加载数据的情况，因此当对 Hbase 进行全量更新时，这是首选方式。

但这种方式在增量加载时，就没有那么高效了，因为新的 HFile 的加入，会触发 Hbase 的 split 和 rebalance 操作，这会使 doBulkLoad 的过程非常缓慢。所以在对 Hbase 进行批量加载的时候，应该尽量使用全量更新的方式，如果增量更新不可避免，则使用原生的 API 接口逐条 put 入库将是最后的选择。

## 2.3 ETL

ETL 是 Extract-Transform-Load 的缩写，是数据的抽取、转换、加载过程，当需要将数据从一个环境转移到另一个环境时（例如从生产环境到分析环境），或者需要对数据进行进一步加工处理时（例如在分析环境中，在数据仓库基础上产出每日交易量指标），即需要借助 ETL 过程。

ETL 是构建数据闭环自循环过程的重要途径，几乎每个环节均可以通过 ETL 来完成。通过专门的 ETL 工具，定制满足业务要求的 ETL 作业，并结合自动调度工具，即可以实现数据的自动循环。



### 2.3.1 ETL 工具

目前国内商用 ETL 工具以 IBM 的 Datastage 为代表, 该 ETL 工具在金融行业有广泛的应用, 但 Datastage 价格昂贵, 许多公司从成本上考虑, 采用了 Pentaho 的开源 ETL 工具 Kettle。

在本书成书时, 尽管 Kettle 本身仍然存在一些 Bug, 但由于部署简单、使用方便并且完全免费的特点, 使其成为越来越多用户的首选。

#### 1. 开源 ETL 工具: Kettle

Kettle 是国外的一款开源 ETL 工具, 中文名称为水壶。在写作本书的时候, 官网上最新的发布版本为 Data Integration 5.4.0, 下载后会得到一个 `pdi-ce-5.4.x.x-xxx.zip` 的压缩包, 使用解压缩工具解压该压缩包, 便可以开始使用 Kettle 了。

Kettle 可以运行在 Windows 环境或者 Linux 环境, 如果运行在 Windows 环境, 则进入解压后的 `data-integration` 目录, 可以看到 `Spoon.bat` (Linux 环境为 `Spoon.sh`), 双击 `Spoon.bat` 打开 Kettle 的图形界面, 如图 2-8 所示。



图 2-8 Kettle 的图形界面

图 2-8 展示了 Kettle 的两个基本组件: 转换 (transformation) 和作业 (job)。转换用来定义数据处理的一个或多个步骤 (step), 如读取文件、过滤输出行、数据清洗、数据加载至目的数据库等。作业用来将多个定制完成的转换串接起来, 使转换能够按照一定的顺序和规则执行。

定义完成的转换和作业可以使用程序或者脚本进行调用, 首先将定义的转换或者作业存储在 Kettle 的资源库 (Repository) 中, 然后通过 Kettle 提供的 Pan 和 Kitchen 组件分别进行调用 (Pan 用来调用 transformation, Kitchen 用来调用 job), 我们将在随后的内容介绍

调用方法。

通常，首次接触“资源库”这个词会让人感觉难以理解，但撇开这个名词本身，它本质上就是创建关系数据库中的一些配置表，这些配置表用来存储转换或者作业的相关信息（如转换的名称、数据库连接的字符串等），Pan 和 Kitchen 组件可以根据资源库里的这些信息来调用对应的转换或者作业。

在使用资源库之前，需要先创建一个资源库，在图形界面中进入工具→资源库→连接资源库，即可以出现相应的创建向导，通过向导可以轻松完成资源库的创建。

一旦成功创建并连接了资源库，随后进行的文件读取和保存操作就可以对应到该资源库。例如，作者在自己的测试环境中连接资源库后，单击菜单文件→打开，Kettle 会自动读取资源库中保存的转换和作业信息，并将所有的转换和作业展示出来以供选择，如图 2-9 所示。



图 2-9 Kettle 资源库中的转换和作业

这些存放在资源库中的转换和作业，可以通过 Pan 和 Kitchen 组件进行调用，Pan 是 Kettle 提供的用于批量调用转换的工具，Kitchen 是用于批量调用作业的工具。在 data-integration 目录中可以找到 Pan.bat 和 Kitchen.bat（Windows 环境对应 Pan.sh，Linux 环境对应 Kitchen.sh）。

在 Linux 环境中可以使用如代码清单 2-37 所示的脚本调用转换。

代码清单 2-37

```
pan.sh -rep=kettle_rep_test -trans="batch-into-table" -dir=/ -user=admin
-pass=admin
-level=Basic
```

调用作业可以使用如代码清单 2-38 所示的命令。

代码清单 2-38

```
kitchen.sh -rep=kettle_rep_test -job="hive-oracle-test" -dir=/ -user=admin
-pass=admin
-level=Basic
```

如果有很多转换和作业需要运行,那么可以将这些命令写在一个 shell 脚本中,然后通过 Linux 系统自带的 Crontab 进行调度,或者通过专门的调度工具进行调度(请参阅 2.4 节)。

Kettle 方面的推荐书籍:《Pentaho Kettle 解决方案:使用 PDI 构建开源 ETL 解决方案》(作者:Matt Casters、Roland Bouman、Jos van Dongen 著,初建军、曹雪梅译,电子工业出版社)。

## 2. 商用 ETL 工具:DataStage

DataStage 是 IBM InfoSphere 开发的一款商用 ETL 工具,是 IBM InfoSphere Information Server 套件的简称。该套件包含三个组件:InfoSphere DataStage and QualityStage Designer、InfoSphere DataStage and QualityStage Director、InfoSphere DataStage and QualityStage Administrator。

InfoSphere DataStage and QualityStage Designer 用于创建 DataStage 作业。

InfoSphere DataStage and QualityStage Director 用于验证、调度、运行和监视 DataStage 作业。

InfoSphere DataStage and QualityStage Administrator 用于系统管理(例如设置 IBM InfoSphere Information Server 用户,记录、创建和移动项目,设置清除记录的条件等)。

显然,相对于 Kettle 的轻量级部署,DataStage 本身的架构已经非常复杂,相应的部署要求也比较高。DataStage 作为 IBM 公司的一款产品,其目标客户群为大型企业,它甚至支持在大型机上运行 ETL 作业(能够生成可在大型机上运行的 COBOL 代码)。由于国内的银行仍然沿用 IBM 公司的大机系统,所以 DataStage 在国内的客户多存在于金融行业。

DataStage 价格昂贵,一般需要支付年服务费、购买 License 等。同样,由于收取年服务费,所以能够提供很好的培训和技术支持。因此,需要根据企业自身特点选择商用 ETL 工具或者开源 ETL 工具,对于小型公司而言,开源工具仍是首选。

### 2.3.2 ETL 作业

ETL 作业是按照一定顺序组织的数据处理过程,它将数据处理的各个环节关联起来,并定义各个环节的触发规则,从而完成整个数据处理流程。

以 Kettle 为例,ETL 作业由多个步骤(或称为作业项)组成,如图 2-10 所示。该作业除了开始的“START”与最后的“成功”步骤外,还包含以下三个实体作业项:

1) 检测昨日交易明细文件是否存在。

2) SQL Server 批量加载。

3) 统计昨日交易。

作业项 1) 负责检测昨日的交易明细文件是否存在, 如果该步骤返回“true”, 则进行下一个作业项, 否则退出作业。

作业项 2) 将昨日的交易明细文件批量加载至 SQL Server 数据库中, 该作业项需要指定文件名称、格式文件等相关信息, 使用的命令即是“SQL Server: bcp in”章节中讲述的方式。如果该作业项执行成功, 那么昨日交易明细数据将增量更新至 SQL Server 的数据库表中; 如果该作业项执行失败, 则退出作业。

作业项 3) 对作业项 2) 中加载的交易数据进行统计, 该作业项执行一段 SQL 脚本, 并将计算结果存储在对应的结果表中。

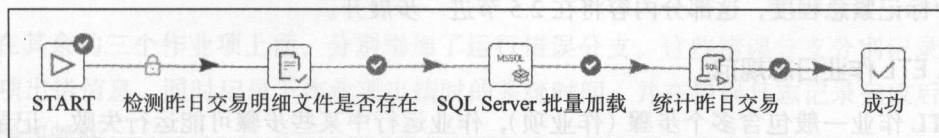


图 2-10 ETL 作业示例

图 2-10 中的 ETL 作业按照预定的顺序将多个作业项串联起来, 完成一个完整的数据加载和统计过程, 该过程的每个步骤作为一个作业项独立存在, 仅当上游的作业项执行成功后, 才开始下一个作业项的执行。

需要注意的是, ETL 工具仅用于作业的创建和简单调度, 如果需要周期性地执行 ETL 作业, 则需要使用专门的调度工具。

为了使 ETL 作业便于调度和监控, 为 ETL 作业制定规范是一项非常重要的工作, 良好的 ETL 作业命名规范和日志规范可以极大地方便作业监控和错误排查。接下来深入介绍这两个实用性的操作规范: ETL 作业命名规范和 ETL 作业日志规范。

### 1. ETL 作业命名规范

ETL 作业命名规范主要是为了通过作业名称来标识作业的归属、重要程度、主要用途等, 以便于作业的自动调度和监控, 它不是 ETL 工具的强制要求。

通常需要根据企业具体的管理要求为 ETL 作业制定命名规范, 该规范要尽可能地反应作业的归属用途等, 并且长度不能太长, 下面给出一个范例:

[ 员工编号 ]. [ 作业类型 ]. [ 作业描述 ]

该命名规范包含三个部分, 用“.”分割:

1) 员工编号, 用于描述作业的归属, 一般使用作业创建人或者负责人的员工编号。

2) 作业类型, 用于描述作业的重要程度, 比如将作业类型定义为 analysis、report、product 等, 分别对应分析、报表、生产。不同的作业类型的作业出现错误时, 可以根据重要程度进行不同等级的报警通知。



3) 作业描述, 用于描述作业的主要功能, 比如图 2-10 中的作业可以描述为 `trx_load_and_static`, 或者使用中文描述 (如果 ETL 工具支持中文名称)。

命名规范同样可以规定 ETL 作业中出现的字母统一使用大写或者小写, 本书采用小写的方式。按照这个规范, 图 2-10 中的 ETL 作业将命名为: `z06837.analysis.trx_load_and_static`, 其中 `z06837` 是员工编号, `analysis` 说明该作业属于分析型的作业。

按照规范进行作业命名后, 作业监控进程便可以自动发现运行失败的作业, 并且根据作业名称中的员工编号找到该员工的邮箱地址和手机号码 (需预先在数据库中保存员工编号与邮箱地址和手机号码的对应关系), 并发送邮件通知和短信提醒, 还可以根据作业类型在邮件中标记紧急程度, 这部分内容将在 2.5 节进一步展开。

## 2. ETL 作业日志规范

ETL 作业一般包含多个步骤 (作业项), 作业运行中某些步骤可能运行失败, 记录下失败原因对于错误排查非常重要。

虽然 ETL 工具都自带日志记录功能, 但系统自动记录的日志信息一般可读性很差且缺乏灵活性。ETL 作业日志规范就是要自定义一个统一且灵活的日志记录方式, 以便于作业的监控和错误排查。下面给出一个 ETL 作业日志规范的范例:

- 1) ETL 作业中需包含记录作业开始和作业完成的作业项。
- 2) 每个作业项均需增加作业项运行失败分支, 并发送邮件通知。
- 3) 日志记录统一记录在数据库表 `etl_job_log` 中。
- 4) 日志记录中的状态在作业状态表 `etl_job_status` 中统一定义。
- 5) 使用统一的存储过程进行日志记录。

根据规范 1、2 的要求, 图 2-10 中的 ETL 作业将修改为图 2-11 所示的样子。

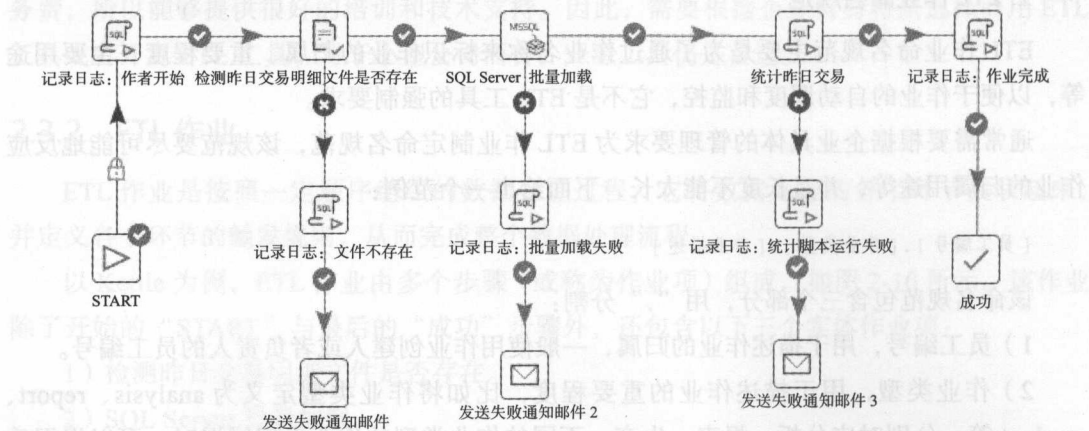


图 2-11 满足规范 1 和规范 2 的 ETL 作业

图 2-11 中,作业开始后增加了一个作业项“记录日志:作业开始”,这个作业项往 etl\_job\_log 表中插入一条新记录,记录今日该作业的开始时间等相关信息。另外,在作业的最后加入了作业项“记录日志:作业完成”,用于更新作业的最终状态,图 2-12 是表 etl\_job\_log 中记录的部分 ETL 作业日志。

id	job_name	run_date	start_time	end_time	upt_time	job_status	remark
4	z06837.analysis.trx_load_and_static	2015-10-09	2015-10-09 01:10:00	(Null)	2015-10-09 01:32:00	2	SQL Server批量加载:出错
3	z06837.analysis.trx_load_and_static	2015-10-08	2015-10-08 01:10:00	2015-10-08 01:40:38	2015-10-08 01:40:38	1	作业成功执行
2	z06837.analysis.trx_load_and_static	2015-10-07	2015-10-07 01:10:00	2015-10-07 01:44:06	2015-10-07 01:44:07	1	作业成功执行
1	z06837.analysis.trx_load_and_static	2015-10-06	2015-10-06 01:10:00	2015-10-06 01:42:35	2015-10-06 01:42:35	1	作业成功执行

图 2-12 作业日志表中的部分记录

在其余的三个作业项上面,分别增加了运行错误分支。这些错误分支分别记录对应的作业项出错信息,同时记录下作业项出错时的系统时间,并在错误日志记录完成后,发送失败通知邮件。

修改后的 ETL 作业会在运行过程中将作业状态自动记录到数据库中,随后 BI 工具可以根据数据库中的日志记录展示监控报表或者进行错误报警。

日志记录表 etl\_job\_log 创建表脚本如代码清单 2-39 所示。

代码清单 2-39

```
create table etl_job_log
(
  id                bigint                not null auto_increment comment '自增长 id',
  job_name          varchar(100)         not null comment '作业名称',
  run_date          varchar(20)          comment '运行日期',
  start_time        datetime             comment '作业开始时间',
  end_time          datetime             comment '作业结束时间',
  upt_time          datetime             default current_timestamp comment '更新时间',
  job_status        int                  not null comment '作业状态 id',
  remark            varchar(1000)        comment '作业状态补充说明',
  primary key (id)
);
```

其中,remark 字段的记录原则为:作业项名称+错误说明,例如,“SQL Server 批量加载:出错。”可以方便追踪到作业出错的作业项。

job\_status 字段是表 etl\_job\_status 的外键,记录的是状态 id,其对应的状态描述可以通过关联表 etl\_job\_status 得到。

作业状态表 etl\_job\_status 创建表脚本如代码清单 2-40 所示。

代码清单 2-40

```
create table etl_job_status
(
```

```

id          bigint          not null auto_increment comment '自增长 id',
status_desc varchar(1000)  not null comment '状态描述',
primary key (id)
);

```

## 2.4 作业调度

调度工具用来对作业进行调度，通过 ETL 工具创建的作业如果需要周期性运行，就需要使用调度工具来完成。调度工具是一个相对复杂的系统，尤其是在跨操作系统、跨应用平台的作业环境中更是如此。

在复杂的作业环境中，需要使用商用调度工具，目前国内使用较多的商用调度工具为 Control-M。该工具是 BMC Software 提供的企业级集中作业调度管理解决方案，能够集中管理跨平台、跨应用的生产控制和调度过程，因此适用于大型复杂的 ETL 调度场景。

一些相对简单的调度场景可以不使用专门的调度工具实现。比如，如果公司的作业环境全部是 Linux 系统，则可以使用系统自带的 crontab 进行调度。

例如，图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，那么可以将代码清单 2-41 的内容保存在文件 run\_trx\_load\_and\_static.sh 中。

代码清单 2-41

```

kitchen.sh -rep=kettle_rep_test -job="z06837. analysis.trx_load_and_static"
-dir=/ -user=admin -pass=admin -level=Basic

```

然后编辑 Linux 系统的 crontab 文件（通过 crontab -e 命令），再在该文件中加入如代码清单 2-42 所示的内容。

代码清单 2-42

```

10 01 * * * /home/queziyang/shell/run_trx_load_and_static.sh

```

crontab 会在每天凌晨 1 点 10 分运行指定的脚本 run\_trx\_load\_and\_static.sh，这样便可以完成每日定时调度该 ETL 作业。显然，使用 crontab 调度作业，本身是没有作业日志记录的，这也是为什么需要遵循 ETL 作业日志规范的原因之一。

## 2.5 监控和预警

监控和预警存在于数据闭环的各个阶段，在所有的自动执行环节均可以植入监控和预警点。前期对 ETL 所做的规范，现在是体现其应用价值的时候了。可以利用这些满足规范

的日志记录进行自动监控和预警。

如果有专职的运维人员负责作业的运行监控，使用专门的监控工具，运维工程师可以监控各个服务器的运行信息，并通过监控工具发送预警邮件。

如果仅仅是 ETL 工程师负责监控自己的 ETL 作业，那么可以使用 BI 工具进行监控和预警。

### 2.5.1 使用监控工具进行监控

有众多的开源监控工具可供使用，如 Zipkin、Ramona、zabbix、Ganglia、Nagios 等，这些监控工具提供了许多定制的监控和预警服务，但它们通常比较偏于底层日志，如 Zabbix 主要用来监控 CPU 负荷、内存使用、磁盘使用、网络状况、端口监视和日志监视。这些监控信息对于保证数据环境的健康运行至关重要，可以根据 CPU 负荷、内存和磁盘的使用情况进行预警，比如在 CPU 负荷持续达到 90% 时进行预警，或者在磁盘使用 90% 时进行预警等。

监控工具专注于系统可用性方面的监控，如果要专注于 ETL 作业的运行情况，那么可以使用 BI 报表工具进行监控。

### 2.5.2 使用 BI 工具进行监控

BI (business intelligence) 工具是企业环境中广泛使用的数据可视化工具，它可提供丰富的数据可视化能力，同时可提供短信、邮件等通知服务。

鉴于数据的监控和预警本身是基于日志信息的，因此可以使用 BI 工具丰富的展示和通知服务进行数据系统的监控和预警。

基于图 2-12 中的 ETL 作业日志表，BI 工具可以定制图形化监控报表，并以 Web 页面的形式展示出来。作业负责人或者运营人员可以登录该 BI 系统，查看监控相应的页面，便可以监控作业是否正常。

例如，近期表现抢眼的 BI 工具 Tableau，可以设置每 15 分钟扫描一下 ETL 作业日志表，一旦发现作业异常，就自动发送邮件通知作业负责人。

通过 BI 工具实现 ETL 作业的监控和预警，这种方式可以推广到整个数据闭环，其图形化的界面让监控变得简单明了。

## 2.6 本章小结

本章围绕数据体系提出了数据闭环的概念，对数据闭环的特征进行了描述，并且进一步介绍了构建数据闭环所涉及的方法和技术。



这里着重介绍了在数据闭环中扮演重要作用的“数据缓冲区”的设立理念和实现方式。通过设立数据缓冲区，可以实现系统解耦，让数据闭环具备良好的扩展性，让公司组织间职责更加分明，使数据环境更加安全等。

数据缓冲区中涉及的数据批量导出/导入技术，需要使用到各个 RDMS 系统的批量操作命令。另外在大数据平台中，需要使用 hadoop shell 和 hive shell 脚本来实现批量操作，对于 Hbase，则提供了 Java 实现的 bulk load 批量导入方式。

ETL 作业为数据闭环中定义数据流转方式的环节。为了实现 ETL 作业的自动化运行和监控，需要引入 ETL 作业规范：命名规范和日志规范。

作业调度则是保证 ETL 作业能够实现自动化的手段，监控和预警则进一步保证了 ETL 作业能够正常运行。

第 3 章将根据这两章提出的数据理念，通过实战的方式完成数据闭环中关键环节的构建。

例如，图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。

图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。

图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。

## 2.5 监控和预警

图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。图 2-11 中的作业配置为每日凌晨 1 点 10 分开始运行，图 2-11 的内容保存在文件 `run_trs_load_and_stat.sh` 中。

### 第3章 Chapter 3

## 实战：打造数据闭环

道生一，一生二，二生三，三生万物。万物负阴而抱阳，冲气以为和。

——老子《道德经》

你不能两次踏进同一条河流，因为新的水不断地流过你的身旁。

——柏拉图《泰阿泰德》

前面两章已经介绍了数据闭环的各个主要环节和涉及的技术，本章将基于前文提出的理念和技术来实现数据闭环。按照数据的流转过程以及主要负责人的职责，整个数据闭环将由三个环节组成，如图 3-1 所示。

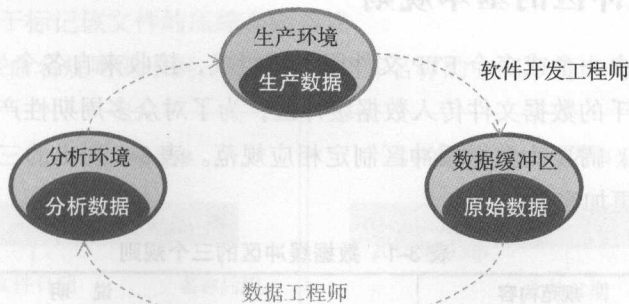


图 3-1 数据流转过程及主要负责人职责示意图

1) 生产数据→原始数据。这个环节的主要负责人为生产系统的软件开发工程师，软件

开发工程师按照数据抽取需求，将生产数据转换为原始数据，并将原始数据按照要求传至数据缓冲区。

2) 原始数据→分析数据。这个环节的主要负责人为数据团队的数据工程师，该节点实际上是解决分析数据的入口问题，是数据闭环中的关键环节。

3) 分析数据→生产数据。该环节的负责人为数据工程师和软件开发工程师，需要两者合作，将分析数据转化为生产数据，供生产系统使用，是数据系统化应用的过程。

在上述第一个环节，开发团队接收来自数据团队的数据抽取需求，在需求中需要明确定义原始数据的格式要求和更新方式等，并且需要产出的数据文件满足文件命名规则。软件开发工程师根据数据需求周期性产出数据文件，并自动上传至数据缓冲区。

在第二个环节，数据团队需要按照一定的规则将数据文件批量加载到数据平台中。该环节是数据闭环中的关键，也是全部由数据团队负责完成的环节。大量原始数据文件已经存放于数据缓冲区中，保证大量数据高效且稳定地流入分析系统，显然无法通过人力完成，因此该环节的系统化实现是本章的重点内容。

在第三个环节，负责人为数据工程师和软件开发工程师。数据工程师负责数据预处理、提供数据接口、将数据模型系统化等，在此过程中，需要软件开发工程师的积极配合。该环节是数据由分析环境进入生产环境的环节，是数据应用于生产系统的过程，本书在应用篇将着重介绍数据的系统化应用过程。

以下将围绕第二个环节详细展开，该环节的关键在于大量数据文件的自动加载。为了实现该目标，需要先制定数据缓冲区的基本规则，然后设计文件自动加载的基本流程，并在此基础上使用 Java 多线程实现数据的高效加载。

### 3.1 数据缓冲区的基本规则

数据缓冲区由一个或多个 FTP 文件服务器组成，接收来自各个生产系统的原始数据，每天会有成百上千的数据文件传入数据缓冲区，为了对众多周期性产生的数据文件进行管理和系统化处理，需要为数据缓冲区制定相应规范。表 3-1 列出的三个规则可以让程序的自动化处理变得更加容易。

表 3-1 数据缓冲区的三个规则

序 号	规范内容	说 明
1	文件存储规则	按照统一的存储方式对文件进行存储
2	文件命名规则	通过文件名称自动匹配更新规则
3	文件清理规则	定期自动清理文件，保证数据缓冲区有足够的可用空间

数据的自动加载将依赖于这三个规则，数据缓冲区的上下游只需遵守这三个规则即可

通过增加配置信息，实现数据的自动对接。为了使规则具有可操作性，下面对这三个规则分别进行细化。

### 3.1.1 文件存储规则

文件存储规则为数据缓冲区定义了数据文件的存储方式，例如，规定传入数据缓冲区中的文件必须使用 .zip 的格式进行压缩。该规则简单易行，且至少具有以下显而易见的优点。

1) 压缩文件可以节省存储空间。当数据文件众多时，压缩文件可以大量节省存储空间，从而提升数据缓冲区的存储利用率。

2) 压缩文件在传输时节省带宽。压缩文件占用更少字节，这在内网带宽紧张时尤其有用。

3) 统一的压缩方式方便程序自动解压。统一使用某种压缩格式，便于后续程序的自动解压，因为程序无需考虑多种压缩格式。

文件存储规则确定后，所有上传数据缓冲区的文件都需要遵守该规则，否则将被监护程序删除。

### 3.1.2 文件命名规则

文件命名规则是表 3-1 三个规则中最重要的一个，因为程序需要根据文件名称识别该文件的后续处理流程。例如，我们给出如下的文件命名规则范例：

1) 文件名称由“名称前缀 + 文件日期 + 中间字符串 + 名称后缀”构成。

2) 中间字符串可以为空。

3) 名称前缀 + 中间字符串唯一标示一个文件。

4) 文件日期标示同一个文件的不同批次。

5) 名称后缀用于标记该文件的压缩存储格式。

图 3-2 是符合该命名规则的示例，其中一个文件名称的中间字符串为空。



图 3-2 文件命名规则示例



3.1.3 文件清理规则

文件清理规则是为了保证数据缓冲区的可用空间而制定的，满足清理规则的数据文件将会被程序自动清理，以腾出空间为新数据文件使用。

通常数据缓冲区会保留同一个文件前缀的不同文件日期的数据文件，如图 3-3 所示。

01-cdt-trx-dtl-20150812-000000.tsv.zip	2015/8/13 19:49	WinRAR ZIP 压缩文件	5,716 KB
01-cdt-trx-dtl-20150813-000000.tsv.zip	2015/8/14 17:30	WinRAR ZIP 压缩文件	5,414 KB
01-cdt-trx-dtl-20150814-000000.tsv.zip	2015/8/15 17:29	WinRAR ZIP 压缩文件	3,651 KB
01-cdt-trx-dtl-20150815-000000.tsv.zip	2015/8/16 17:30	WinRAR ZIP 压缩文件	6,140 KB
01-cdt-trx-dtl-20150816-000000.tsv.zip	2015/8/17 17:30	WinRAR ZIP 压缩文件	4,238 KB
01-cdt-trx-dtl-20150817-000000.tsv.zip	2015/8/18 17:29	WinRAR ZIP 压缩文件	5,267 KB
01-cdt-trx-dtl-20150818-000000.tsv.zip	2015/8/19 17:29	WinRAR ZIP 压缩文件	5,126 KB
01-cdt-trx-dtl-20150819-000000.tsv.zip	2015/8/20 17:30	WinRAR ZIP 压缩文件	4,689 KB

图 3-3 数据缓冲区中的文件列表（部分）

数据缓冲区中的文件如果不进行清理，则会耗尽磁盘空间，导致数据缓冲区无法继续提供服务。清理规则可能非常简单，例如，每日 23:00 自动清除一个月前的数据文件。

根据上述清理规则，每天在 23:00 时，程序将根据数据缓冲区中文件名称的文件日期，自动删除一个月前的历史数据，腾出的存储空间将用于次日凌晨接收新的数据文件。

3.2 自动加载的流程

既然已经明确了数据缓冲区的基本规则，那么我们认为一个满足规则要求的数据缓冲区已经搭建起来了（可以向 IT 运维人员提出搭建 FTP 文件服务器的需求），接下来真正进入数据自动加载的设计过程。

首先，仔细分析文件从数据缓冲区自动加载到分析环境中的流程，可以将这个过程细分为 4 个阶段：扫描文件、下载文件、解压文件和加载文件，如图 3-4 所示。

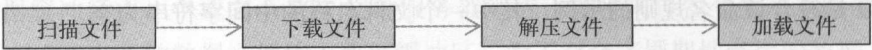


图 3-4 文件自动加载流程分解

这 4 个阶段的功能描述如下。

- 1) 扫描文件。该阶段用于判断当前数据日期的文件是否已经完整存在于数据缓冲区中，一旦数据文件完整存在（而不是部分）于数据缓冲区，就在数据库中标记该文件已存在。
- 2) 下载文件。将数据库中记录的已经存在（且尚未下载）的文件从数据缓冲区下载到分析环境中。
- 3) 解压文件。在分析环境中，将已经下载完成的数据文件进行解压缩。解压后的文件

便于后续的加载操作。

4) 加载文件。将解压后的文件批量加载至数据库表中。

上述4个阶段，数据文件会根据处理进程发生状态改变，这个自动加载流程将会涉及12种文件状态，图3-5展示了状态之间的转化。

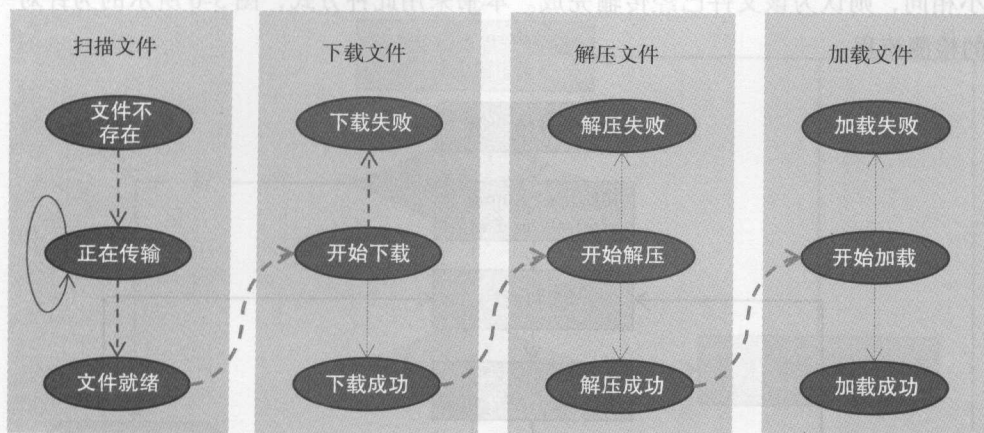


图 3-5 文件状态转化图

在扫描文件的开始阶段，处于“文件不存在”状态，一旦程序扫描到该文件，状态就变为“正在传输”，处在这个状态的文件仍处于FTP传输过程中，因此可能不完整。

一旦文件处于“文件就绪”状态，就说明该文件已经完整存在于数据缓冲区，可以进入下载文件阶段。处于“下载成功”状态的文件，可以进入解压文件阶段，而“解压成功”状态的文件将进入加载文件阶段。

### 3.2.1 扫描文件

通常，生产系统周期性（每月、每天或每小时）产出原始数据，并上传至数据缓冲区。自动加载程序需要及时发现新上传至数据缓冲区中的数据文件，这需要一个程序进程对数据缓冲区的文件目录进行监控，为便于描述，我们称这个监控进程为扫描文件进程。

从图3-5中可知，扫描文件进程需要记录下文件的当前状态，所以需要有一个数据库表保存所有文件的当前状态，另外还需要一个配置表来告诉程序哪些文件名称需要扫描，以及需要监控的文件目录等信息，这些内容将在数据库设计章节详细介绍。

现在已经有了存储配置信息的数据库表，还需要解决一个关键问题：如何判断数据缓冲区中的文件已经完整上传？因为数据文件是通过FTP方式由生产环境上传到数据缓冲区中的，一些文件可能有数百兆字节，整个传输过程可能需要几分钟至数十分钟，因此需要判断文件是否传输完成。有两种简单方式可以判断文件是否传输完成。

1) 文件传输过程中, 在文件名称之后添加标记后缀, 表示该文件尚未传输完成, 一旦 FTP 传输完成, 就将文件名称后面的标记后缀去除。这样, 带有标记后缀的文件即为传输尚未结束的文件。

2) 通过多次检测文件的大小来判定该文件是否传输完成。比如, 连续 5 次检测到文件的大小相同, 则认为该文件已经传输完成。本书采用此种方式, 图 3-6 所示的为针对一个文件的检测流程。

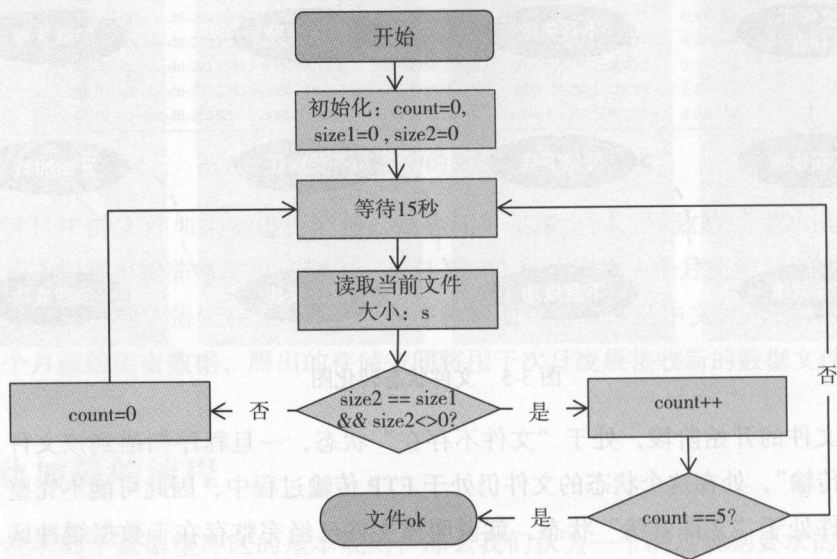


图 3-6 多次检测法判断是否文件完整

多次检测法的主要思想是通过多次检测一个文件的大小来判断该文件是否已经传输完成, 当连续多次检测到文件的大小时, 则认为文件已经传输完成 (否则文件大小应该改变)。根据这个思想, 图 3-6 中设置了一个计数器 `count`, 初始值为 0, 每隔 15 秒检测一下文件大小, 当连续多次 (图 3-6 中为 5 次) 检测到文件大小相同时, 进程结束, 文件状态变为“文件就绪”。

### 3.2.2 下载文件

扫描文件进程在数据库中将文件状态变更为“文件就绪”, 下载文件进程将这些处于“文件就绪”状态的文件下载到分析环境的对应服务器中。

同样, 下载文件进程也要根据情况更新数据库中文件的状态, 比如“开始下载”“下载成功”“下载失败”等, 并且当文件下载成功后, 要记录文件所在的路径, 以便随后的解压进程找到对应的文件。

为了提高效率，下载文件采用多线程方式，通过调节并行线程数来寻求下载效率和带宽压力之间的平衡点。图 3-7 展示了多线程下载文件的流程。

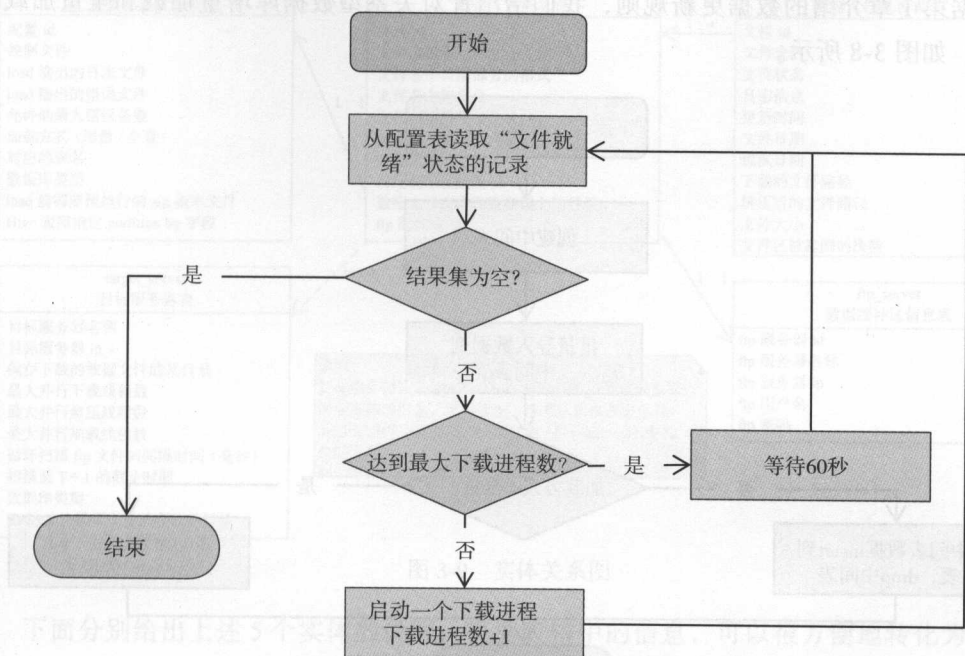


图 3-7 多线程下载文件流程图

### 3.2.3 解压文件

解压文件进程将文件状态为“下载成功”的文件进行解压缩，它根据下载文件进程记录的文件路径找到对应的压缩文件，然后根据文件后缀调用对应的解压缩接口。解压缩结束后，将文件状态变更为“解压成功”或者“解压失败”，当文件“解压成功”后，需要记录解压之后的文件存放路径，以便随后的加载文件进程使用。

解压缩可以使用 Java 自带的解压缩程序包进行，也可以通过调用命令行的方式实现（绝大部分操作系统都支持 `unzip` 命令，尽管其运行后的返回值略有不同）。

### 3.2.4 加载文件

加载文件进程负责将状态为“解压成功”的文件批量加载到对应的表中，目标表可以是关系型数据库，也可以是大数据平台的 Hive 表。

加载文件进程需要一些附加信息来确定文件的加载方式（增量或者全量）、对应的目标表信息等。这些信息通过配置表提供，将在数据库设计章节详述。



最后，加载文件进程调用批量导入命令（参阅 2.2.4 节）将数据文件快速高效地导入目标表中。

根据第 1 章介绍的数据更新规则，我们给出针对关系型数据库增量加载和全量加载的流程图，如图 3-8 所示。

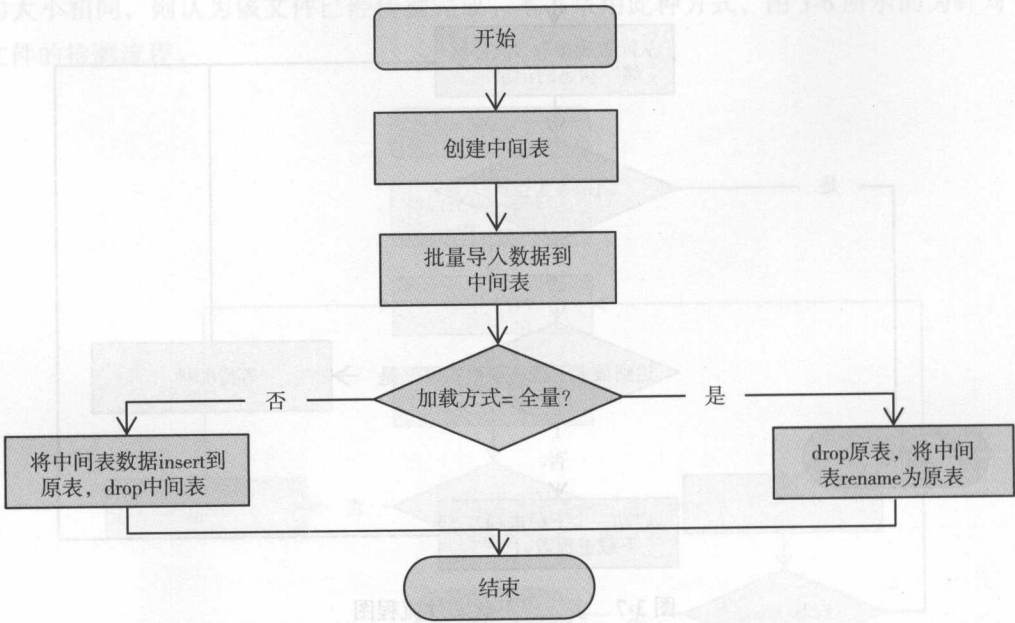


图 3-8 增量加载和全量加载流程（关系型数据库）

### 3.3 自动加载程序的数据库设计

根据之前的设计，自动加载程序需要从数据库配置表中获取配置信息，并不断更新相关的状态，表 3-2 列出了自动加载程序需要的所有配置表。

表 3-2 自动加载程序的配置表

表 名	中文名称	用 途
file_settings	数据文件信息表	存储数据文件名称、日期等配置信息
file_status	数据文件状态表	存储数据文件的状态
load_config	加载配置信息表	存储数据库中表的相关信息
ftp_server	数据缓冲区信息表	存储数据缓冲区文件服务器的相关信息
target_server	目标服务器表	存储目标服务器的相关信息
file_targetServer_rel	文件 - 目标服务器关联表	file_settings 与 target_server 多对多映射中间表

本书使用 Hibernate 进行关系数据库映射，上述配置表除 file\_targetServer\_rel 表外，其

余表均对应一个实体 (entity)，图 3-9 给出了这些实体间的关系图。

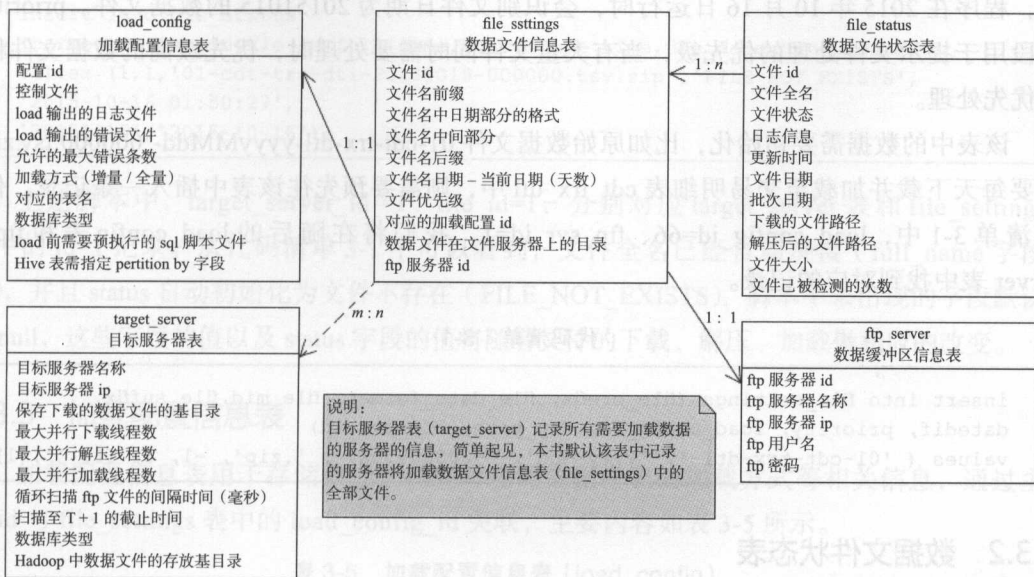


图 3-9 实体关系图

下面分别给出上述 5 个实体的说明, 根据表格中的信息, 可以很方便地转化为所需的 ddl 建表脚本。

### 3.3.1 数据文件信息表

数据文件信息表的主要内容如表 3-3 所示。

表 3-3 数据文件信息表 (file\_settings)

字 段	数据类型	注 释
id	bigint	自增长 id, 主键
file_prefix	varchar(100)	文件名前缀
file_date_format	varchar(20)	文件日期格式, 必须是 Java 的 SimpleDateFormat 类支持的格式
file_mid	varchar(100)	文件名中间部分
file_suffix	varchar(10)	文件名后缀
datedif	int	文件日期 - 当前日期 (天数)
priority	int	文件优先级。数值越大, 优先级越低
load_config_id	bigint	对应的文件导入配置 id、load_config 外键
ftp_path	varchar(100)	文件在 ftp 服务器上的目录
ftp_svr_id	bigint	ftp 服务器 id, ftp_server 外键

文件的全名通过字段 file\_prefix、file\_mid、file\_suffix 以及文件日期拼接而成 (参阅 3.1.2

节)。表中 datedif 字段的作用是为了让程序自动识别当日的数据文件，比如当 datedif=-1 时，程序在 2015 年 10 月 16 日运行时，会识别文件日期为 20151015 的数据文件。priority 字段用于提示文件处理的优先级，当有大量文件同时需要处理时，优先级高的数据文件将会优先处理。

该表中的数据需要初始化，比如原始数据文件 01-cdt-trx-dtl-yyyyMMdd-000000.tsv.zip 需要每天下载并加载至交易明细表 cdt\_trx\_dtl 中，则需要预先往该表中插入一条记录。代码清单 3-1 中，load\_config\_id=66、ftp\_svr\_id=1，我们将在随后的 load\_config 表和 ftp\_server 表中找到对应的记录。

代码清单 3-1

```
insert into file_settings (file_prefix, file_date_format, file_mid, file_suffix,
datedif, priority, load_config_id, ftp_path, ftp_svr_id)
values ( '01-cdt-trx-dtl-', 'yyyyMMdd', '-000000.tsv', '.zip', -1, 1, 66, '', 1);
```

### 3.3.2 数据文件状态表

自动加载程序会根据数据文件信息表中的相关信息以及程序运行的日期，自动插入文件状态记录，并保存在数据文件状态表中，主要内容如表 3-4 所示。

表 3-4 数据文件状态表 (file\_status)

字 段	数据类型	注 释
id	bigint	自增长 id，主键
target_server_id	bigint	目标服务器 id
file_id	bigint	文件 id，file_settings 外键
full_name	varchar(120)	文件全名
status	varchar(50)	文件状态
log_info	varchar(2000)	日志信息
upt_time	datetime(0)	更新时间
file_date	varchar(20)	文件日期
batch_date	varchar(20)	批次日期
file_path	varchar(100)	下载的文件路径
unzip_file_path	varchar(200)	解压后的文件路径
file_size	bigint	文件大小 (Bytes)
file_size_check_cnt	tinyint	文件已经被检测的次数
priority	int	处理优先级

例如，当程序在 2015 年 10 月 16 日运行时，程序根据 file\_settings 表中的相关配置信息，会自动往该表中插入一条信息（为方便表述，将程序自动执行的 sql 脚本列出，如代码清单 3-2 所示）。

代码清单 3-2

```
insert into file_status
(target_server_id,file_id,full_name,status,upt_time,file_date,batch_date)
values (1,1,'01-cdt-trx-dtl-20151015-000000.tsv.zip','FILE_NOT_EXISTS',
'2015-10-16 01:50:27',
'20151015','2015-10-16');
```

上述脚本中，target\_server\_id=1、file\_id=1，分别对应 target\_server 表和 file\_settings 表中的 id=1 记录。从代码清单 3-1 中可以看到，文件全名已经自动拼接（full\_name 字段值），并且 status 自动初始化为文件不存在（FILE\_NOT\_EXISTS），脚本中未出现的字段默认为 null，这些字段的值以及 status 字段的值将随着文件的下载、解压、加载做相应的改变。

### 3.3.3 加载配置信息表

加载配置信息表用于存储数据文件到对应的数据库表的加载方式等相关信息，通过主键 id 与 file\_settings 表中的 load\_config\_id 关联，主要内容如表 3-5 所示。

表 3-5 加载配置信息表 (load\_config)

字 段	数据类型	注 释
id	bigint	自增长 id，主键
control_file	varchar(100)	控制文件，含路径（关系型数据库适用）
log_file_path	varchar(100)	load 输出的日志文件路径（关系型数据库适用）
bad_file_path	varchar(100)	load 输出的错误文件路径（关系型数据库适用）
max_errors	int	允许的最大错误条数（关系型数据库适用）
load_type	varchar(10)	加载方式：增量 / 全量
tab_nam	varchar(50)	对应的表名
db_type	varchar(20)	数据库类型
pre_exec_sql	varchar(100)	load 前需要预执行的 sql 脚本文件，含路径
partition_by_col	varchar(50)	Hive 表需指定 partition by 字段

表 file\_settings 中，load\_config\_id=66，则在该表中有以下记录：

id	control_file	log_file	bad_file	max_errors	load_type	tab_nam	db_type	pre_exec_sql	partition_by_col
66			(Null)	10000000	added	adobe.cdt_trx_dtl	hive	(Null)	LOAD_DAY

以上记录说明 file\_settings 中 id=1 的文件对应了表 adobe.cdt\_trx\_dtl，并且该表是一张 Hive 表，其分区字段是 LOAD\_DAY。

load\_config 表中的部分字段仅适用于关系型数据库，如 control\_file、log\_file\_path、bad\_file\_path、max\_errors，这些字段用于组成关系型数据库批量加载命令。



### 3.3.4 数据缓冲区信息表

数据缓冲区信息表用于存储 ftp 文件服务器的相关信息，主要内容如表 3-6 所示。

表 3-6 数据缓冲区信息表 (ftp\_server)

字 段	数据类型	注 释
id	bigint	自增长 id, 主键
ftp_host	varchar(100)	ftp 服务器 ip
ftp_user	varchar(50)	ftp 用户名
ftp_pwd	varchar(50)	ftp 密码
ftp_nam	varchar(50)	ftp 名称

表 file\_settings 中, ftp\_server\_id=1 的记录在本表中的信息如下 (ftp\_host 字段和 ftp\_pwd 字段进行了屏蔽, 生产上应该进行加密存储)。

id	ftp_host	ftp_user	ftp_pwd	ftp_nam
1	10.10.10.3	hadoop	Hadoop_1234567890	hadoop-ftp

### 3.3.5 目标服务器表

目标服务器表是 file\_settings 与 target\_server 多对多映射中间表，主要内容如表 3-7 所示。

表 3-7 目标服务器表 (target\_server)

字 段	数据类型	注 释
id	bigint	自增长 id，主键
hostname	varchar(100)	目标服务器名称
ip	varchar(20)	目标服务器 ip
localBaseDir	varchar(100)	保存下载的数据文件的基目录
paraFtpCnt	int	最大并行下载线程数
paraUnZipCnt	int	最大并行解压线程数
paraLoadCnt	int	最大并行加载线程数
ftpLoopInterval	bigint	循环扫描 ftp 文件的间隔时间（毫秒）
stopScanTime	varchar(50)	扫描至 T+1 的截止时间，即如果截至 T+1 的 10:00:00 仍未检测到文件，则认为文件不存在
db_type	varchar(20)	数据库类型
hdfsBaseDir	varchar(100)	Hadoop 中数据文件的存放基目录，当 db_type=hive 时需指定

target\_server 表中，localBaseDir 配置项记录了保存文件的本地基目录，即文件现在只在本地后，将存放在该基目录中，自动加载程序在下载文件时，会根据该文件对应的表名和文件日期在该基目录下创建相应的子目录，并将数据文件保存在对应的子目录中。

paralFtpCnt 配置项指明了程序可以同时运行的最大下载线程数，paralLoadCnt、paralUnZipCnt 分别指明了并行加载的最大线程个数和并行解压的最大线程个数。

ftpLoopInterval 配置项用于声明扫描 ftp 文件服务器的间隔时间，10000 表示每 10 秒扫描一次 FTP 文件服务器。

hdfsBaseDir 用于 Hive 加载时指明 HDFS 文件存放的路径。

### 3.4 自动加载程序的多线程实现

本书中的自动加载程序使用 Java+Hibernate 实现，需要首先准备项目所需要的 hibernate 和 annotation 相应版本的 jar 包。

程序需要初始化文件状态表，即根据 file\_status 表中的配置，一次性产生当日需要处理的文件的初始记录，这些记录会插入 file\_status 表中，供随后的处理模块使用。

为了便于并行处理，我们将自动加载程序的整个流程划分为多个子项目：

- 1) 扫描文件子项目，取名为 ScanFiles。
- 2) 将下载文件和解压文件归为一个 Java 项目，取名为 DownLoadAndUnZip。
- 3) 加载文件为另一个子项目，例如针对 Oracle，可以创建一个 LoadToOracle 的项目，针对加载至 Hive 表中的场景，可以创建一个 LoadToHive 的项目，当然也可以放在一起用参数区分。

这些 Java 子项目之间通过 file\_status 配置表进行数据通信，图 3-10 展示了这些子项目的信息交互。

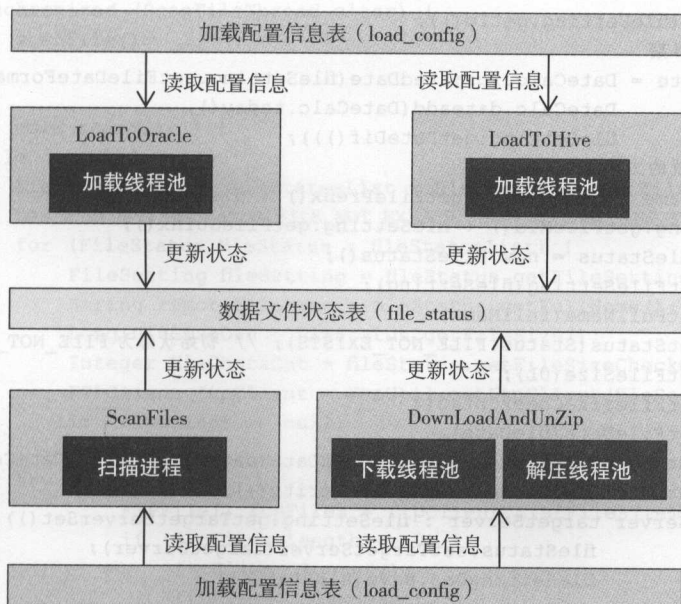


图 3-10 项目之间的关联关系

### 3.4.1 ScanFiles

ScanFiles 是常驻内存进程，包含初始化文件状态和扫描文件两个功能。初始化文件状态模块在每天凌晨根据数据文件信息表（file\_settings），将当日需要处理的数据文件信息初始化至数据文件状态表（file\_status）中，初始化字段包含文件全名、批次日期、文件日期、文件状态（初始化为 FILE\_NOT\_EXISTS）等，更多内容可参考前面对数据文件状态表的介绍。

扫描文件模块根据数据文件状态表中当日的初始化信息，在对应的目录中扫描数据文件，更新对应的状态信息。

代码清单 3-3 和代码清单 3-4 分别给出了初始化文件状态表的代码和扫描文件的代码。

代码清单 3-3

```
private static void generateTheList() {
    if (fileStatusDao.getTodayStatusCnt().longValue() > 0L) {
        // 设置 TaskDay 为 today
        TaskDayStr = DateCalc.formatedDate(dateFormatStr, DateCalc.today());
        return;
    }

    FileSettingDao fileSettingDao = new FileSettingDaoImpl();
    List<FileSetting> fileSettings = fileSettingDao.getAllFileSettings();
    Set<Long> fileIdSet = new HashSet<Long>();
    for (FileSetting fileSetting : fileSettings) {
        if (fileIdSet.contains(fileSetting.getId()))
            continue;
        else
            fileIdSet.add(fileSetting.getId());
        // 文件名中的日期
        String fileDate = DateCalc.formatedDate(fileSetting.getFileDateFormat(),
            DateCalc.dateadd(DateCalc.today(),
                fileSetting.getDateDif()));
        // 当天需要下载的文件全名，不含路径
        String fullName = fileSetting.getFilePrefix() + fileDate +
            fileSetting.getFileMid() + fileSetting.getFileSuffix();
        FileStatus fileStatus = new FileStatus();
        fileStatus.setFileSetting(fileSetting);
        fileStatus.setFullName(fullName);
        fileStatus.setStatus(Status.FILE_NOT_EXISTS); // 初始状态为 FILE_NOT_EXISTS
        fileStatus.setFileSize(0L);
        fileStatus.setFileSizeCheckCnt(0);
        fileStatus.setFileDate(fileDate);
        fileStatus.setBatchDate(DateCalc.formatedDate(dateFormatStr, DateCalc.today()));
        fileStatus.setPriority(fileSetting.getPriority());
        for (TargetServer targetServer : fileSetting.getTargetServerSet()) {
            fileStatus.setTargetServer(targetServer);
            fileSettingDao.save(fileStatus);
        }
    }
    try {
```

```

Thread.sleep(300); // 暂停0.3秒
} catch (InterruptedException e) {
    logger.error(e.toString());
}
}
// 设置TaskDay为今天
TaskDayStr = DateCalc.formatedDate(dateFormatStr, DateCalc.today());
}

```

generateTheList() 方法通过 fileStatusDao.getTodayStatusCnt(). longValue() > 0L 判断当天的文件状态是否已经初始化，如果已经初始化，则返回；否则通过 FileSettingDao 遍历所有的 file\_settings 记录，并通过 FileStatus 实体对象将每条 file\_settings 记录初始化为一个 fileStatus 对象，最后通过 fileSettingDao.save(fileStatus) 将对应的初始化文件状态记录保存至 file\_status 表中。

generateTheList() 方法仅在每天凌晨成功调用一次即可，调用成功之后，file\_status 表中将会插入当日需要处理的所有数据文件信息，并且文件的初始状态为 FILE\_NOT\_EXISTS。

#### 代码清单 3-4

```

public class ScanFileThread extends Thread {
    private static Logger logger = Logger.getLogger(ScanFileThread.class);
    private static FileStatusDao fileStatusDao = new FileStatusDaoImpl();
    public void run() {
        synchronized (ScanFileThread.class) {
            scanFile();
        }
    }
    private void scanFile() {
        while (true) {
            List<FileStatus> fileStatusList = fileStatusDao.getFileByStatus(
                new Status[] { Status.FILE_NOT_EXISTS, Status.CHECKING_FTP_FILE });
            for (FileStatus fileStatus : fileStatusList) {
                FileSetting fileSetting = fileStatus.getFileSetting();
                String remoteFileName = fileStatus.getFullName(); // 文件全名
                Long fileSizeOld = fileStatus.getFileSize();
                Integer fileCheckCnt = fileStatus.getFileSizeCheckCnt();
                FTPClient ftpClient = FtpUtil.getFtpClient(fileSetting);
                if (ftpClient == null)
                    continue;
                try {
                    FTPFile[] ftpFiles = ftpClient.listFiles(remoteFileName);
                    if (ftpFiles.length == 0) {
                        ftpClient.logout();
                        ftpClient.disconnect();
                        continue;
                    }
                }
            }
        }
    }
}

```



```

3.4.1 ScanFiles
}
if (fileStatus.getStatus().equals(
Status.FILE_NOT_EXISTS.name())) {
fileStatus.setStatus(Status.CHECKING_FTP_FILE);
} else if (fileStatus.getStatus().equals(Status.
CHECKING_FTP_FILE.name())) {
Long fileSizeNew = ftpFiles[0].getSize();
if (fileSizeNew.longValue() != fileSizeOld.
longValue()) {
fileStatus.setFileSize(fileSizeNew);
fileStatus.setFileSizeCheckCnt(0);
} else {
if (fileCheckCnt < 5)
fileStatus.setFileSizeCheckCnt(fileCheckCnt + 1);
else
fileStatus.setStatus(Status.FTP_FILE_OK);
}
}
fileStatusDao.update(fileStatus);
} catch (IOException e) {
logger.error("扫描FTP文件时出错: " + e.toString());
} finally {
if (ftpClient.isConnected()) {
try {
ftpClient.logout();
ftpClient.disconnect();
} catch (IOException e) {
logger.error("断开FTP连接时出错: " + e.
toString());
}
}
}
}
try {
Thread.sleep(15000); // sleep 15 秒
} catch (InterruptedException e) {
logger.error("Thread.sleep(15000) 时出错: " + e.toString());
}
}
}
}

```

ScanFileThread 类使用了 FileStatusDao 类用于数据库的读/写功能，该类的父类 BaseDao 提供了 Hibernate 的 SessionFactory 以及事务上的支持。

ScanFileThread 类的主要方法 scanFile() 中用于读取文件类型为 Status.FILE\_NOT\_EXISTS 以及 Status.CHECKING\_FTP\_FILE 的全部记录，通过 ftpClient.listFiles(remoteFileName) 查看指定文件是否存在于 FTP 服务器中，一旦发现文件存在，就将文件的状态由 Status.

FILE\_NOT\_EXISTS 更新为 Status.CHECKING\_FTP\_FILE。

随后程序通过多次（每次间隔 15 秒）比较文件大小来确定 FTP 服务器上的文件是否已经上传完成，当连续 5 次检测到的文件大小都相同时，则认为文件已经完整上传至 FTP 服务器，程序将该文件的状态由 Status.CHECKING\_FTP\_FILE 更新为 Status.FTP\_FILE\_OK，标明该文件可以开始下载了。

### 3.4.2 DownloadAndUnZip

在 DownloadAndUnZip 项目中，包含两个重要的 Java 类：GetFileThread 类和 UnZipThread 类，它们分别负责下载文件和解压文件。

GetFileThread 类的主要作用是采用多线程的方式下载文件，其核心方法是代码清单 3-5 中所示的 ftpGetFile 方法。

代码清单 3-5

```
private static void ftpGetFile(FileStatus fileStatus) {
    FTPClient ftpClient = FtpUtil.getFtpClient(fileStatus.getFileSetting());
    FileSetting fileSetting = fileStatus.getFileSetting();
    String remoteFileName = fileStatus.getFullName(); // 文件全名，带路径
    if (fileStatus.getStatus().equalsIgnoreCase("FTP_FILE_OK")) {
        fileStatus.setStatus(Status.DOWNLOAD_START); // 更新下载状态
        fileStatusDao.update(fileStatus);
        String localBaseDir = globalConfig.get("localBaseDir").getConfigContent();
        String tableName = fileSetting.getLoadConfig().getTableName();
        String ftpPath = fileSetting.getFtpPath();
        Date fileDate = DateCalc.dateadd(DateCalc.today(), fileSetting.getDateDif());
        String fileDateStr = DateCalc.formatedDate(fileSetting.
            getFileDateFormat(), fileDate);
        String localDir = null; // 文件下载后，存放的本地目录
        if (ftpPath.equals(""))
            localDir = localBaseDir + "/" + tableName + "/" + fileDateStr;
        else
            localDir = localBaseDir + "/" + tableName + "/" + ftpPath +
                "/" + fileDateStr;
        // 创建指定路径
        File ldir = new File(localDir);
        if (!ldir.exists())
            ldir.mkdirs();
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(ldir.getAbsolutePath() + "/" +
                remoteFileName);
        } catch (FileNotFoundException e) {
            logger.error("FileOutputStream: " + e.toString());
        }
    }
    try {
```

```

ftpClient.setFileType(FTPClient.BINARY_FILE_TYPE);
ftpClient.setBufferSize(1024000);
String downloadFilePath = null;
if (ftpClient.retrieveFile(remoteFileName, fos)) { // 下载完成
    downloadFilePath = localDir + "/" + fileStatus.getFullName();
    fos.close();
    fileStatus.setFilePath(downloadFilePath);
    fileStatus.setStatus(Status.DOWNLOAD_FINISHED);
    fileStatusDao.update(fileStatus);
} else {
    fileStatus.setStatus(Status.DOWNLOAD_FAILED); // 更新下载状态
    fileStatusDao.update(fileStatus);
}
} catch (IOException e) {
    fileStatus.setStatus(Status.DOWNLOAD_FAILED); // 更新下载状态
    fileStatus.setLogInfo(e.toString());
    fileStatusDao.update(fileStatus);
    logger.error("下载文件出错: " + e.toString());
} finally {
    if (ftpClient != null)
        try {
            ftpClient.logout();
            ftpClient.disconnect();
        } catch (IOException e) {
            logger.error("ftpClient.disconnect() 出错: " + e.toString());
        }
}
}
}

```

该方法用于检测状态为 FTP\_FILE\_OK 的数据文件，根据配置信息自动创建以文件对应表名、FTP 远程路径和文件日期拼接的本地文件存放路径（localDir = localBaseDir + "/" + tableName + "/" + ftpPath + "/" + fileDateStr;），然后将文件下载至上述创建的目录中。在下载过程中，同时更新文件的下载状态，并记录文件下载完成后的保存路径。

UnZipThread 类的作用是将文件状态为 DOWNLOAD\_FINISHED 且需要解压缩的数据文件（通过文件后缀区分）进行解压，并记录解压后的文件名称和路径。代码清单 3-6 展示了 UnZipThread 类的主要方法 unzip()。

代码清单 3-6

```

public static void unzip(FileStatus fileStatus) {
    if (!fileStatus.getStatus().equals("DOWNLOAD_FINISHED")) {
        logger.error("File Status not yet in DOWNLOAD_FINISHED!");
        return;
    }
    FileSetting fileSetting = fileStatus.getFileSetting();

```

```

String fileSuffix = fileSetting.getFileSuffix().toLowerCase();
fileStatus.setStatus(Status.UNZIP_START);
fileStatus.setUpdateTime(new Timestamp(new Date().getTime()));
fileStatusDao.update(fileStatus);
if (fileSuffix.equals(".zip"))
    zip(fileStatus);
else if (fileSuffix.equals(".gz"))
    gz(fileStatus);
}

```

unzip() 方法支持 .zip 和 .gz 两种压缩方式，如果压缩文件是其他压缩方式，比如 .gzip，那么需要自行编写针对 .gzip 格式文件的解压代码。

unzip() 方法调用成功后，会根据实际解压结果更新文件状态为 UNZIP\_FAILED 或者 UNZIP\_FINISHED，如果解压成功，则记录解压后的文件名称及文件路径。

### 3.4.3 LoadToHive

LoadToHive 用于将 load\_config 中 db\_type='hive' 的数据文件导入对应的 Hive 表中，并且根据 load\_type 字段进行不同的加载处理。代码清单 3-7 展示了 LoadToHive 的核心代码。

代码清单 3-7

```

private void loadToTable(TargetServer targetServer, FileStatus fileStatus) {
    fileStatus.setStatus(Status.LOAD_START);
    fileStatusDao.update(fileStatus);
    String loadType = fileStatus.getFileSetting().
        getLoadConfig().getLoadType();

    String hdfsBaseDir = targetServer.getHdfsBaseDir();
    String hdfsDir = null;
    if (loadType.equalsIgnoreCase("added"))
        hdfsDir = hdfsBaseDir + "/" +
            fileStatus.getFileSetting().getLoadConfig().getTableName() + "/" +
            fileStatus.getFileDate();
    else if (loadType.equalsIgnoreCase("covered"))
        hdfsDir = hdfsBaseDir + "/" +
            fileStatus.getFileSetting().getLoadConfig().getTableName();
    if (!AddToHive.createHdfsDir(fileStatus, hdfsDir)) // 创建 hdfs 目录
        return;
    if (loadType.equalsIgnoreCase("added")) { // 增量，将新的 partition 加入 Hive 表中
        if (!AddToHive.addPartitionToHive(fileStatus, hdfsDir))
            return;
    } else if (loadType.equalsIgnoreCase("covered")) { // 全量，将 Hive 表中的历史数据删除
        if (!AddToHive.delOldDataOfHive(fileStatus, hdfsDir))
            return;
    }
}

```



```
AddToHive.copyDataToHdfs(fileStatus, hdfsDir);// 将 Linux 本地文件复制到 hdfs 目录
}
```

LoadToHive 首先创建 hdfs 目录，然后判断加载方式，如果加载方式是“added”（增量），则在 hive 表中增加一个 partition（该 partition 指向之前创建的 hdfs 目录），再将数据文件复制到 hdfs 目录，完成 Hive 表增量更新的过程；如果是“covered”（全量）加载方式，则先将历史数据删除，然后将新的数据文件复制到 hdfs 目录即可。

### 3.4.4 LoadToOracle

LoadToOracle 是自动加载程序在 Oracle 数据库上的实现。与 Hive 数据仓库的不同之处在于，Oracle 数据库（以及其他关系型数据库）有其自身的 DBMS 体系，因此其数据加载方式一般采用其自身提供的实用工具，LoadToOracle 调用 Oracle 的 sqldr 进行数据批量加载（sqldr 请参考第 2 章关于 Oracle: sqldr 的内容）。

LoadToOracle 与 LoadToHive 的另一个主要不同点在于，数据文件在加载到目标数据库表之前，会先创建一个中间表（在原表名后加上 temp），数据文件先加载至 temp 表中，然后再分别根据增量或者全量的加载方式分别处理（参考 1.2.2 节中关于数据更新规则的内容）。

代码清单 3-8 给出了 LoadToOracle 的核心代码。

代码清单 3-8

```
private void loadToTable(TargetServer targetServer, FileStatus fileStatus) {
    fileStatus.setStatus(Status.LOAD_START);
    fileStatusDao.update(fileStatus);
    String loadType = fileStatus.getFileSetting().getLoadConfig().getLoadType();
    String tableName = fileStatus.getFileSetting().getLoadConfig().getTableName();
    String createTempTableRes = OracleUtil.createTempTable(tableName);// 创建 temp 表
    if (!createTempTableRes.equals("")) {
        fileStatus.setStatus(Status.LOAD_FAILED);
        fileStatus.setLogInfo(createTempTableRes);
        fileStatusDao.update(fileStatus);
        return;
    }
    if (!loadIntoTempTable(fileStatus)) { // 加载数据至 temp 表
        fileStatus.setStatus(Status.LOAD_FAILED);
        fileStatusDao.update(fileStatus);
        return;
    }
    if (loadType.equalsIgnoreCase("added")) { // 增量更新
        // temp 表中数据 insert 到原表，并将 temp 表 drop
        if (OracleUtil.insertAndDropTable(tableName))
            fileStatus.setStatus(Status.LOAD_FINISHED);
        else
    }
```

```

        fileStatus.setStatus(Status.LOAD_FAILED);
        fileStatusDao.update(fileStatus);
    } else if (loadType.equalsIgnoreCase("covered")) { // 全量更新
        OracleUtil.dropAndRenameTable(tableName); //drop 原表并将 temp 表重命名为原表
    }
}

```

当数据文件较大时，批量加载耗时较长，体现在代码上，loadIntoTempTable(fileStatus) 将花费大量时间等待 sqllldr 批量加载命令返回（可能达 1 个小时以上，取决于数据文件的大小以及 Oracle 数据库服务器的硬件资源），这个过程中，file\_status 表中的文件状态一直不发生改变，这会让人有些担心。

一个改进的办法是，在等待 sqllldr 批量加载命令返回的过程中，通过 count 对应 temp 表中数据的条数来反馈加载进程，可以每隔 5 分钟 count 一次 temp 表，并将对应的行数记录下来，这样就可以直观感受到加载程序是否正在正常工作。

### 3.4.5 自动加载程序的部署架构

至此，我们已经完成了自动加载程序的设计和实现过程，现在看一下自动加载程序的部署框架。

将自动加载程序的主要组成项目 ScanFiles、DownloadAndUnZip 和加载（LoadToOracle、LoadToHive）导出为可执行 jar 包，得到 ScanFiles.jar、DownloadAndUnZip.jar、LoadToOracle.jar、LoadToHive.jar。这些 jar 包各自专注于完成自己的功能，要完成自动加载程序的整个过程，需要这些可执行 jar 包之间相互协调工作，图 3-11 展示了这些 jar 包的部署架构。

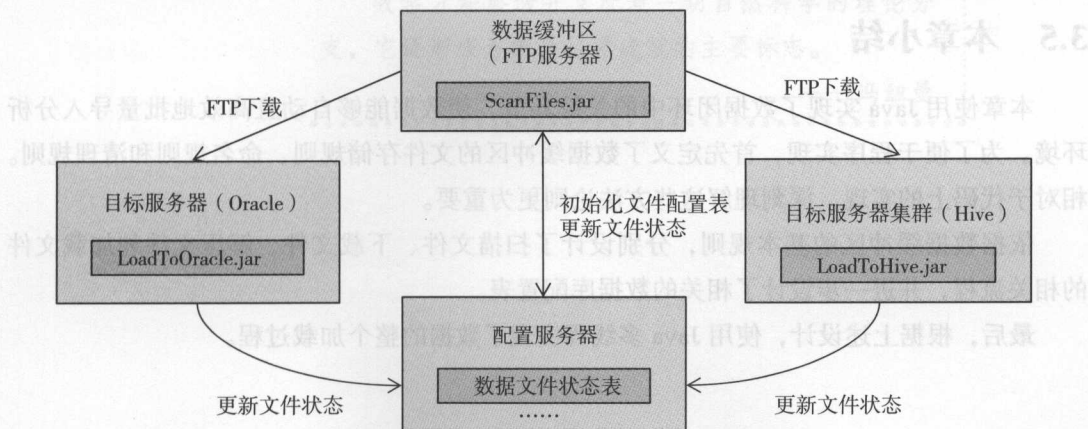


图 3-11 自动加载程序的部署架构图

如图 3-11 所示，将可执行 jar 文件复制到相应的服务器上，然后通过 java-jar 命令启动

这些可执行 jar 文件即完成部署。例如在 Linux 服务器上,通过执行命令“java-jar/home/queziyang/jars/ScanFiles-1.0.jar &”,即在后台启动 ScanFiles 进程,通过 jps 命令可以查看对应的进程信息。

数据缓冲区上需要部署 ScanFiles.jar,所有后续进程均依赖于 ScanFiles.jar 的产出。

每个目标服务器(或服务器集群)上都需要部署 DownLoadAndUnZip.jar,它将对应的数据文件从 FTP 服务器上下载至目标服务器,并在目标服务器上完成文件解压。注意到,文件的解压缩并没有在 FTP 服务器上进行,而是下载至本地后解压,这样做是很有必要的,因为文件解压需要占用大量 CPU 和 I/O 资源,因此在 FTP 服务器上执行这项任务会严重影响数据缓冲区的正常功能。

### 3.4.6 程序的维护和优化

数据文件在目标服务器上解压后,自动加载程序并未将文件删除。为了节省服务器的存储空间,定期删除本地服务器上保留的数据文件是非常必要的。可以通过批处理的方式定期删除历史数据文件,也可以在 LoadToOracle 或者 LoadToHive 中加入删除数据文件的功能,即当数据文件成功加载后,将原始数据文件以及解压后的数据文件删除。

另一个优化点是可以将上述配置数据库的过程可视化,即在上述 Java 项目的基础上包装成 Java Web 项目,这样往 file\_settings 表、load\_config 表以及其他表中插入数据时,可以直接在页面上完成,并且每个数据文件的配置信息、日志文件信息等均可以在页面上查询展示,这样便于运维人员进行系统维护,而不需要直接操作后台数据库。

## 3.5 本章小结

本章使用 Java 实现了数据闭环中的关键环节,使数据能够自动且高效地批量导入分析环境。为了便于程序实现,首先定义了数据缓冲区的文件存储规则、命名规则和清理规则。相对于代码上的实现,深刻理解这些方法论则更为重要。

依据数据缓冲区的基本规则,分别设计了扫描文件、下载文件、解压文件和加载文件的相关流程,并进一步设计了相关的数据库配置表。

最后,根据上述设计,使用 Java 多线程实现了数据的整个加载过程。

## 第二部分 Part 2

# 分析篇 (Analytics)

工欲善其事，必先利其器。

——《论语·卫灵公》

数学方法渗透并支配着一切自然科学的理论分支，它逐渐成为衡量科学成就的主要标志。

——冯纽曼

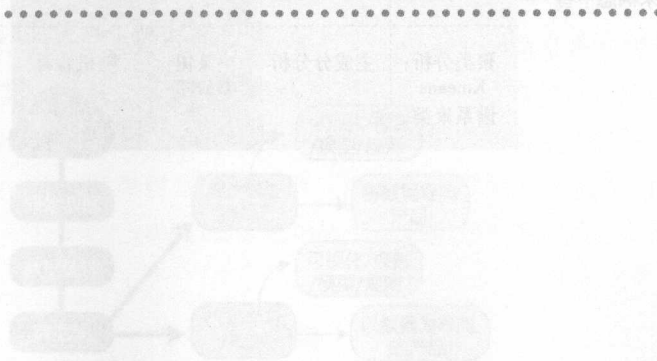


图 2-1 数据分析过程

数据预处理包括两部分：其一，对数据记录的处理，主要包括缺失值、重复记录、错



这些可执行 jar 文件即完成部署。例如在 Linux 服务器上，通过执行命令“java -jar/home/queziyang/jars/ScanFiles-1.0.jar &”，即在后台启动 ScanFiles 进程，通过 jps 命令可以查看对应的进程信息。

数据缓冲区上需要部署 ScanFiles.jar，所有后续进程均依赖于 ScanFiles.jar 的启动。

分析方法，是大数据的价值核心，使数据变为知识。对于数据分析师来说，任何可以对数据进行计算、分析以发现规律、提炼规则、获取知识的方法和算法都可以为我所用，包括统计学、社会学、运筹学、市场营销、数据挖掘、自然语言处理及机器学习等。本文结合银行和互联网企业的实际应用场景，介绍一些最具代表性的数据分析方法。

关于数据分析方法和算法的划分，五花八门。数据分析应该以业务问题为导向，首先要熟悉业务及行业知识，其次要有形成清晰的分析思路，最后才是方法与工具。为了方法和工具而分析，往往只能自娱自乐。这里我们从解决业务问题的角度出发，将最常用的分析方法与算法划归为 3 类，即关联类、预测类和描述类。后面将结合案例介绍其中最常用的 10 种方法。

关联类 重点在“隐含的关系”，针对症状寻根问原，发现症状背后的蛛丝马迹	社交网络分析		Logistic 回归		预测类 重点在“为什么”以及“将会发生什么”，从对规律的提炼进而预见未来
	关联规则：Apriori 序列关联	对应分析	时间序列	线性回归	
描述类 重点在“是什么”，通过望闻问切来洞悉一些症状	文本分析	数据可视化	人工神经网络	贝叶斯算法	
	聚类分析：Kmeans 谱系聚类	主成分分析	决策树：CART	随机森林	

最后，根据上述设计，使用 Java 多线程实现了数据的整个加载过程。

## 数据预处理

南方有鸟焉，名曰蒙鸠，以羽为巢，而编之以发，系之苇苕，风至苕折，卵破子死。巢非不完也，所系者然也。西方有木焉，名曰射干，茎长四寸，生于高山之上，而临百仞之渊，木茎非能长也，所立者然也。蓬生麻中，不扶而直；白沙在涅，与之俱黑。

——《荀子·劝学》

欲事之无繁，则必劳于始而逸于终。

——苏轼《决壅蔽》

数据之于数据分析，好比食材之于烹饪，砖瓦之于高楼，其质量是否可靠，处理是否得当，将直接决定数据分析的结论是否准确可靠。在整个数据分析过程中，数据的清洗处理占据相当大的工作比重（如图 4-1 所示）。

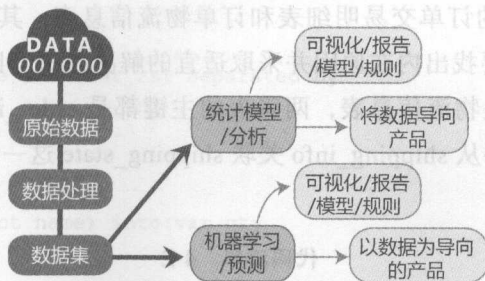


图 4-1 数据分析过程

数据预处理包括两部分：其一，对数据记录的处理，主要包括缺失值、重复记录、噪

声数据（例如明显超出正常范围或错误的数 据，年龄=200、ATM 取现金额低于 100）、极值（通常依据接下来的分析要求，例如为了提高聚类效果而对特别大的数值进行处理）等的处理；其二，对变量的处理，主要包括降维、变量衍生、剔除共线性、剔除无关变量等，变量处理通常取决于接下来要做什么分析，例如聚类需要降维，而回归分析则不需要。本章将针对这些典型问题介绍数据预处理的方法。

## 4.1 数据表的预处理

大多数情况下，我们都是从数据仓库中提取数据，首先接触到的就是数据表，哪怕是大型企业的数 据表，也可能或多或少存在各种各样的问题，所以对数据表的检查和处理就显得非常重要。常见的数据表存在两个问题：主键不唯一和数据缺失。

主键不唯一，也就是有重复数据。最简单的检验方法就是对比主键的数量和表中数据的数量。假如表 table01 的主键为 key，则可以使用代码清单 4-1 来对比 key\_cnt 与 data\_cnt 的取值是否一致。

代码清单 4-1

```
select
    count(distinct key) as key_cnt
    ,count(*) as data_cnt
from table01
```

如果取值一致，则说明主键唯一，之后可以放心地与其他数据表进行关联；如果取值不一致，则需要拿出部分样例、业务及技术进行讨论，最终找出问题所在并确定解决方案。比如笔者曾遇到的情况是，订单表的订单号有重复，后来发现是由订单的重复支付引起的，最终的解决方案是订单取最后一次支付的信息。

数据缺失并不是指变量有空值，而是指使用到的同样维度的数据表之间的观测数量不一致，比如同一统计期的订单交易明细表和订单物流信息表，其订单数量应该是一致的，如果发现不一致，则需要找出内在原因并采取适宜的解决办法。比如，order\_detail 是订单信息表，shipping\_info 是物流信息表，两张表的主键都是 order\_id，已经确认 order\_detail 是没有问题的，目前需要从 shipping\_info 关联 shipping\_state 这一字段，可以使用代码清单 4-2 所示的 SQL 代码。

代码清单 4-2

```
select
    A.*
    ,B.shipping_state
```

```
,B.order_id
from order_detail A
left join shipping_info B on A.order_id=B.order_id
```

使用 left join 是因为担心 shipping\_info 有缺失，从而导致订单信息丢失。另外，关联 B.order\_id 的原因是，可以根据这一字段为空的情况来判断 shipping\_info 的订单缺失情况。

数据处理的过程，也就是宽表形成的过程，在其中的每一步都检查中间表的观测数量，也是发现以上两种问题的常用方法之一。

## 4.2 变量的预处理

变量的预处理一般包含两块内容：一是缺失值的处理，二是极值的处理。

### 4.2.1 缺失值的处理

#### 1. 了解哪些变量存在缺失

对于缺失值，首先要了解总体情况，哪些变量存在缺失，缺失率如何？一般情况下，缺失率超过 20% 的变量在分析中应考虑予以剔除。代码清单 4-3 给出的 SAS 代码可以计算数据表中每一个变量的缺失情况，其中 temp01 表示原始表。

代码清单 4-3

```
data _null_;
  set temp01 nobs=total;
  call symput('total',total);
  stop;
run;

proc contents noprint data=temp01 out=temp02;
run;

proc sql noprint;
  select distinct name into:var_lst separated by ' '
  from temp02;
quit;

proc sql noprint;
  select count(distinct name) into:var_qty
  from temp02;
quit;

proc sql noprint;
  select distinct name into:var1 -:%cmpres(var%eval(&var_qty+0))
```



```

from temp02;
quit;

%macro missing();
data temp03;
    set temp01 nobs=nobs;
    %do i=1 %to &var_qty;
        if missing(&&var&i) then do;
            m_&&var&i...+1;
        end;
        mr_&&var&i=m_&&var&i/&total;
        keep mr_&&var&i;
    %end;
    if _N_=nobs then output;
run;
%mend;
%missing;

proc transpose data=temp03 out=temp04;
run;

```

缺失值的产生，有系统原因和人为原因两种。系统原因是指软硬件的问题导致数据储存失败，而造成数据缺失；人为原因是指因人的操作失误或者故意隐瞒导致的数据缺失，比如数据录入错误，或者市场调查中被访人拒绝回答。

## 2. 缺失值的类型

如何对缺失值进行预处理，取决于缺失值的类型。一般分为以下三类。

- 1) 完全随机缺失，指数据的缺失是随机的，不依赖于任何已观察到的数据和未观察到的数据。
- 2) 随机缺失，指数据的缺失不是完全随机的，即该类数据的缺失依赖于已观察到的数据，而不依赖于未观察到的数据。
- 3) 完全非随机缺失，指数据的缺失依赖于未观察到的数据。

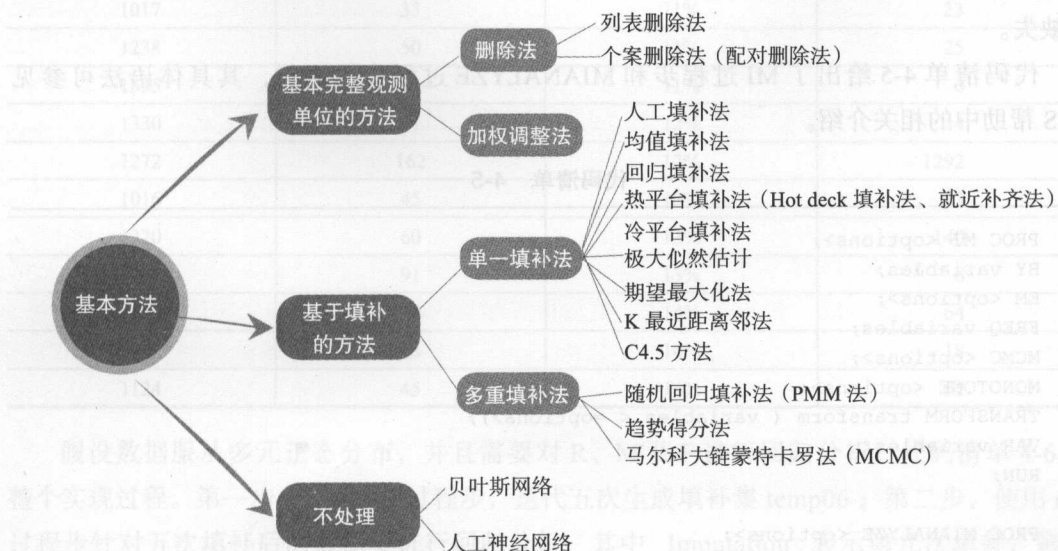
## 3. 缺失值的处理

在了解了变量的缺失情况以及缺失产生的原因和类型后，就要对缺失值进行处理。对于字符型变量，通常不做处理，而是单独作为一类进行分析；对于数值型变量，方法有多种，图 4-2 列出了缺失值处理的一些基本方法，比较常用的有个案删除法、均值补填法和多重填补法中的 MCMC 方法。

### (1) 个案删除法

对于任何存在缺失值的变量，直接删除缺失记录。在缺失值占比非常小（笔者认为在 1% 以下）的情况下，这是一种非常简单有效的方法。然而，该方法以减少样本量来换取数

据的完整性,有很大的局限性,例如,会丢失隐藏在缺失值当中的信息;当变量有成百上千个时,因每个变量的少数缺失值而删除掉的记录,累加起来可能会非常可观。因此,使用该方法需要特别慎重。



## (2) 均值填补法

针对数值型变量,使用平均值来填充该变量的缺失值。使用该方法对缺失值进行填补时,不会影响变量的集中趋势(均值估计不变),但是会造成变量的方差和标准差变小,给强调数据离散趋势的分析造成影响。代码清单 4-4 给出的 SAS 代码可以实现均值填补。

代码清单 4-4

```
proc stdize data=temp01 reponly method=mean out=temp05 ;
var variables;
run;
```

其中,proc stdize 是标准化的过程步,配合 reponly (replace missing data only) 使用可以实现缺失值填补;variables 为需要填补的变量;method 包含 mean 和 median 两个选项,前者指使用均值进行填补,后者指使用中位数进行填补,当数据为偏态分布时,建议使用中位数进行填补。

## (3) 多重填补法

多重填补法由 Rubin 于 1977 年首次提出,现已形成一个比较系统的理论。多重填补法是单一填补法的延伸,不同之处在于对每一个缺失值用多个可能的值填补,以反应缺失值

的不确定性，形成多个完整的数据集，然后用针对完整数据集的统计方法来对每一个填补数据集分别进行统计分析，综合形成最后的结果。SAS 中的 MI 过程步和 MIANALYZE 过程步可以分别实现多次填补和综合分析的功能，其中基于多元正态性假设的马尔科夫链蒙特卡罗方法（Markov Chain Monte Carlo, MCMC）是 MI 过程步的默认选项，主要针对随机缺失。

代码清单 4-5 给出了 MI 过程步和 MIANALYZE 过程步的语句，其具体语法可参见 SAS 帮助中的相关介绍。

代码清单 4-5

```
PROC MI <options>;
BY variables;
EM <options>;
FREQ variables;
MCMC <options>;
MONOTONE <options>;
TRANSFORM transform ( variables < =options>);
VAR variables;
RUN;

PROC MIANALYZE <options>;
BY variables;
CLASS variables;
MODELEFFECTS e_ects;
<label:> TEST <equation1; : : : <equationk>>< =options>;
STDERR variables;
RUN;
```

下面结合实例具体说明多重填补法是如何实现的。表 4-1 是一份某银行公司客户的 R、F、M 数据，其中 R 是客户最近一次有效交易（为银行带来利润的交易）距今天的天数，F 是客户总体活跃度（包括交易、查询、账户管理等），M 是客户带来的累计利润，表名为 temp01。

表 4-1 某银行公司客户的 RFM 数据（样例，数据已经过转化处理）

Csr_ID	M (万)	F	R
1133	50	5%	81
1324	56	6%	11
1026	55	6%	16
1107	48	7%	35
1069	63	7%	14
1010	71	8%	133
1206	70	9%	5

(续)

Csr_ID	M (万)	F	R
1035	36	10%	3
1252	36	11%	6
1017	33	11%	23
1238	50	11%	25
1335	51	11%	0
1330	61	12%	4
1272	162	12%	1292
1016	45	12%	23
1270	60	13%	140
1201	91	13%	6
1321	40	14%	64
1193	68	14%	18
1124	45	14%	52

假设数据服从多元正态分布，并且需要对 R、M 及 F 进行回归分析。代码清单 4-6 为整个实现过程。第一步，使用 MI 过程步，迭代五次生成填补集 temp06；第二步，使用 reg 过程步针对五次填补后的数据集进行回归分析，其中 \_Imputation\_ 表示第几次填补；第三步，使用 mianalyze 过程步，结合填补集回归方程的截距、回归系数进行综合统计，生成总体参数的估计值、95% 的可信区间以及总体参数是否为 0 的 T 检验结果。

代码清单 4-6

```
proc mi data=temp01 seed=1000 nimpute=20 out=temp06;
run;

proc reg data=temp06 outest=temp07 covout noprint;
  model R=M F;
  by _Imputation_;
run;

proc mianalyze data=temp07;
  modeleffects intercept M F;
run;
```

经过估计，Csr\_ID='1188' 客户的 R=290。同样，可以估计 F 和 M 的缺失值。

假如我们只需要填补缺失值，而不需要综合统计，通常有两种做法：一是 nimpute 选项设置为 1，即只进行一次填补；二是使用若干次填补的均值代替缺失值。

总体而言，处理缺失值的方法有很多种，这里只介绍最常用的三种方法。没有哪一种方法是普遍适用的，每种方法都有优缺点，需要结合具体业务和实际情况进行使用，表 4-2



是这三种方法的优缺点比较。另外,在某些情况下,缺失值并不需要单独处理,因为模型本身对缺失值就有一定的处理能力,比如贝叶斯网络和神经网络。

表 4-2 三种缺失值处理方法的优缺点比较

类 型	优 点	缺 点
删除法	简单易行,针对极小缺失比例数据非常有效	减少原始数据,导致信息损失,容易发生数据偏离
单一填补法	常见的处理方式,成本较低	造成方差和标准差变小
多重填补法	产生多个填补值,反映了缺失值的不确定性	成本较高

## 4.2.2 极值的处理

按照形成原因,极值分为两类:第一类是数据问题导致的极大值,比如某些情况下数值型缺失值会被表示成“999999”;第二类是真实存在的极大值,比如使用信用卡购买房屋,则该笔交易金额就非常大。第一类极值通常按缺失值处理,第二类极值是否需要处理以及如何处理,取决于接下来要做什么分析。对于一些分析方法,极值的存在会影响分析的结果,比如聚类分析中的 Kmeans 算法,对极值特别敏感,如不做处理会得出错误的结果。针对这类分析,一般的极值处理方法是使用 99 分位点替代比其更大的数值。代码清单 4-7 给出的 SAS 代码可以实现该功能。

代码清单 4-7

```
proc contents noprint data=temp01 out=temp02;
run;

proc sql noprint;
select distinct name into:var_1st separated by ' '
from temp02;
quit;

proc sql noprint;
select count(distinct name) into:var_qty
from temp02;
quit;

proc sql noprint;
select distinct name into:var1 --:%cmpres(var%eval(&var_qty+0))
from temp02;
quit;

%macro pts99();
%do i=1 %to &var_qty;
proc univariate data=temp01 noprint;
var &&var&i;
output out=work.tmp_&i p99=&&var&i...p99;
```

```

run;

data temp08;
    set temp01;
    if (_n_=1) then set work.tmp_&i;
    if &&var&i>&&var&i..._p99 then &&var&i=&&var&i..._p99;
    drop &&var&i..._p99;
run;
%end;

data temp09;
    merge
%do i=1 %to &var_qty;
        work.tmp_&i
%end;;
run;
%mend;
%pts99;

```

## 4.3 变量的设计

数据中的信息通过变量体现出来,设计合理、组合得当的变量能够以较小的数据量覆盖更多角度的信息,以提升分析结论的含金量。例如,市场研究中的 RFM 模型,通过消费频次、消费金额、最近一次消费日期三个变量就能够准确地衡量客户的购买行为。目前比较主流的评分卡和预测模型都是基于宽表来训练模型,而变量的设计和衍生是构建宽表的关键。可以说同样一批数据训练出来的模型是否稳定、预测是否精准,取决于变量的设计,而不是建模方法。下面介绍两种变量设计和衍生方法。

### 4.3.1 暴力衍生

一般而言,变量越多,提供的信息也越多,从而使数据分析的结果更为全面可信。然而通常情况下,一份原始数据的变量个数非常有限,例如交易明细数据可能只包含卡号、交易流水号、交易日期、交易时间、交易金额、交易类型等几个变量。这就需要进行变量衍生,通过更多的视角解读数据,使变量蕴含的信息更加全面和丰富,暴力衍生法是最常用的方法,通过主体、客体、时间以及统计量四个角度的信息交叉(见图 4-3),甚至直接取笛卡儿积,设计出更多的变量。



图 4-3 暴力衍生的变量设计角度

图 4-4 和图 4-5 给出了信用卡公司和电商公司的变量设计示例。实际工作中,可以视

业务的不同而灵活变通。需要注意的是，变量要有业务含义，不能为了衍生而衍生。另外，变量衍生都可以使用 SAS 或 R 等工具进行编程实现。

主体	客体	时间	统计量
<input type="checkbox"/> 顶层账户 <input type="checkbox"/> 清算层账户	<input type="checkbox"/> 刷卡金额 <input type="checkbox"/> 刷卡笔数 <input type="checkbox"/> 取现金额 <input type="checkbox"/> 取现笔数 <input type="checkbox"/> 分期金额 <input type="checkbox"/> 分期笔数	<input type="checkbox"/> 1个小时 <input type="checkbox"/> 1天 <input type="checkbox"/> 3天 <input type="checkbox"/> 10天 <input type="checkbox"/> 30天 <input type="checkbox"/> 90天	<input type="checkbox"/> Count <input type="checkbox"/> Count distinct <input type="checkbox"/> Sum <input type="checkbox"/> Average <input type="checkbox"/> Max <input type="checkbox"/> Min

图 4-4 信用卡公司的变量设计示例

主体	客体	时间	统计量
<input type="checkbox"/> 账号 <input type="checkbox"/> 卡片 <input type="checkbox"/> 设备 <input type="checkbox"/> IP	<input type="checkbox"/> 交易金额 <input type="checkbox"/> 交易笔数 <input type="checkbox"/> 卡片 <input type="checkbox"/> 设备 <input type="checkbox"/> IP <input type="checkbox"/> 商户	<input type="checkbox"/> 1个小时 <input type="checkbox"/> 1天 <input type="checkbox"/> 3天 <input type="checkbox"/> 10天 <input type="checkbox"/> 30天 <input type="checkbox"/> 90天	<input type="checkbox"/> Count <input type="checkbox"/> Count distinct <input type="checkbox"/> Sum <input type="checkbox"/> Average <input type="checkbox"/> Max <input type="checkbox"/> Min

图 4-5 电商公司的变量设计示例

### 4.3.2 交叉升维

以 Logistic 回归为例，传统的建模思路是降维。一般来说，建模过程如下。

1) 准备训练数据集。这个数据集格式为每行一个客户，该行数据每个字段代表客户的一个维度，如性别、年龄、城市、年收入、近 3 个月交易金额等。假设该数据集每个客户有 50 个维度。

2) 使用传统建模工具，如 SAS 的 EM 模块，将上面的数据集作为输入，运行 Logistic 回归过程。

3) 回归算法依据一些统计量，在 50 个维度的基础上选出部分维度作为模型的变量，并计算出这些变量对应的系数。假设选择 5 个维度（性别、年龄、城市、年收入、近 3 个月交易金额），最后的模型可能为：

$$y = 0.49 + 0.09 * \text{性别} + 0.32 * \text{年龄} + 0.88 * \text{城市} + 0.04 * \text{年收入} + 0.002 * \text{近 3 个月交易金额}$$

可见,为了使模型收敛且尽量避免过拟合等问题,回归算法在输入的50个维度中仅选择了其中的5个维度作为最后输出,其余45个维度信息被扔掉了。

在大数据背景下,传统架构的存储瓶颈和计算瓶颈通过“分布式”得到了解决,数据挖掘模型与大数据思维相结合,产生了一些令人惊奇的运作方式。笔者结合实际应用向大家介绍另外一种建模思路:升维。

所谓升维,就是通过一个数据预处理过程(该预处理过程使用维度离散、维度交叉等方法),在原数据维度基础上衍生出更多的维度。同样以上述数据集为例,其升维方式如下。

1) 维度离散。如上述“城市”维度,通过离散使得每个城市可以成为一个维度:城市\_上海、城市\_北京、城市\_广州……如果数据集中有100个城市,则该维度可以离散为100个维度。

2) 维度交叉。维度交叉是将离散后的两个或多个维度之间进行笛卡尔积。如将离散后的城市维度和离散后的年收入维度进行交叉,则可以得到交叉后的维度:城市\_上海-年收入10000、城市\_上海-年收入50000……如果离散后的城市维度有100个,离散后的年收入维度有50个,则交叉后将得到额外的 $100 \times 50 = 5000$ 个维度。

3) 将上述离散后的维度、离散后交叉的维度与原始维度结合起来,即为最后的升维结果。升维后的维度数量轻易就会达到10万个以上。

为了方便操作,开发了一个数据升维小工具,目前应用在某商业银行中,界面如图4-6所示。该工具包含了维度离散、数据归一化、维度交叉等算法。

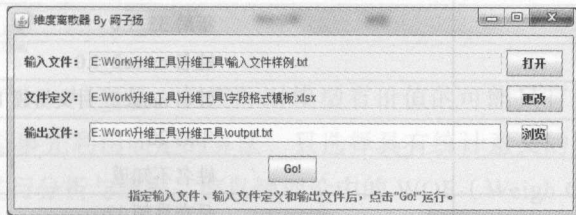


图 4-6 数据升维小工具

对于表4-3的样例数据,应用升维工具处理后将由6个维度衍生出33个维度,如表4-4所示。

表 4-3 升维前的样例数据

姓名	年龄	年收入	月交易额	月交易笔数	达标标记
name1	32	240 000	15 000	15	1
name2	33	200 000	20 000	23	1
name3	20	10 000	8000	16	-1



表 4-4 升维后的维度名称

维度序号	维度名称
1	姓名 name1
2	年龄 32
3	年收入 24000
4	年收入
5	月交易额 13
6	月交易额
7	月交易笔数
8	年龄 32_ 年收入 24000
9	年龄 32_0.0
10	年龄 32_ 月交易额 13
11	姓名 name1_ 年收入 24000
12	姓名 name1_0.0
13	姓名 name1_ 月交易额 13
14	姓名 name2
15	年龄 33
16	月交易额 14
17	年收入 20000
18	年龄 33_ 月交易额 14
19	年龄 33_ 年收入 20000
20	姓名 name2_ 年收入 20000
21	年龄 33_1.0
22	姓名 name2_1.0
23	姓名 name2_ 月交易额 14
24	年龄 20
25	姓名不知道
26	月交易额 12
27	年收入 1000
28	年龄 20_ 月交易额 12
29	年龄 20_ 年收入 1000
30	姓名不知道_ 年收入 1000
31	年龄 20_0.125
32	姓名不知道_ 月交易额 12
33	姓名不知道_0.125

通常，对于包含 10 万个维度的数据集，SAS 的建模工具将无法处理，主要原因如下。

- 1) SAS (或者 R 语言) 处理高维度数据会消耗大量内存，将服务器拖死。
- 2) SAS 回归的过程是降维过程，辛辛苦苦升到 10 万维，经过 SAS 回归后，很可能只

会选择其中的 5 个维度用于建模。

这种情况下,需要使用专门的超高维建模工具,目前要么自己开发,要么使用开源工具。这里推荐一款比较好用的开源工具:Liblinear,该工具由台湾大学的 Chih-Jen Lin 博士开发,感兴趣的读者可以从网上下载该工具。本节的目的是介绍一种新的变量衍生方法,对升维后的数据建模方法不做进一步阐述。

## 4.4 变量筛选

通过变量衍生,我们将得到成千上万的变量,按照辩证法一分为二的看法:其一,变量多,意味着分析时可选择的余地大;其二,对于某个特定的分析,可能只要用到其中的一部分变量,全部使用只会降低计算效率。因此,在分析之前需要结合方法和问题选择变量,仍然是以下两类处理:

- 筛选显著变量:筛选显著变量适用于有监督的学习或有目标的分析,例如分类和回归,通过显著性指标,筛选出对分析目标有较大关系的变量,后面的建模和分析都基于这些变量展开。
- 剔除共线性:变量多意味着更多的共线性问题,简单来说就是某些变量的数值可以通过一组变量计算出来,变量之间的共线性会影响分析结论的可信度和模型的鲁棒性,是很多分析方法都要避免的问题。

### 4.4.1 筛选显著变量

经过变量衍生会产生一堆变量,但实际对模型有价值的可能只有少数几个变量,目前比较常见的处理方法是事先利用固定的算法,只选择具有统计意义的变量,即对目标变量具有显著影响的变量进行分析与建模。信息熵理论中的 WOE (Weigh Of Evidence) 值和 IV (Information Value) 值体现了变量的信息含量以及对目标变量的影响度,常作为筛选显著变量的依据。

关于信息熵理论的内容可以查阅相关资料,这里直接介绍 WOE 和 IV 如何计算。首先要明确的一点是,这两个指标仅适用于类别变量,对于连续变量可以直接计算相关系数。例如,对于某类信用卡分期产品,要分析不同年龄的客户是否存在偏好差异。目标变量是 0/1 类型,1 表示客户接受了产品营销(响应),0 表示没有接受产品营销(未响应);对年龄按照目标做离散化处理,成为分组变量。两个变量间 WOE 和 IV 值的计算公式如表 4-5 所示。最终,各分组的 IV 值加起来即 0.786,就是年龄对是否会响应分期产品营销这个目标的显著性。

表 4-5 WOE 值和 IV 值的计算

分组	响应 客户数	未响应 客户数	响应客户 占比 R1	未响应客户 占比 R0	差值 = R1-R0	WOE= LN(R1/R0)	IV
<22 岁	52	48	26%	6%	20%	1.47	0.293
22 ~ 24 岁	38	62	19%	8%	11%	0.90	0.101
25 ~ 27 岁	27	73	14%	9%	4%	0.39	0.017
28 ~ 30 岁	22	78	11%	10%	1%	0.12	0.002
31 ~ 35 岁	17	83	9%	10%	-2%	-0.20	0.004
36 ~ 40 岁	13	87	7%	11%	-4%	-0.52	0.023
41 ~ 45 岁	10	90	5%	11%	-6%	-0.81	0.051
46 ~ 55 岁	8	92	4%	12%	-8%	-1.06	0.079
55+ 岁	7	93	4%	12%	-8%	-1.20	0.098
未知	6	94	3%	12%	-9%	-1.37	0.119
合计	200	800	100%	100%			0.786

需要注意的是，变量分组应该遵循单调性，且为了保证样本的均匀分布，要求每个区间内的样本量是总样本量的 5% ~ 30%。随着响应率的增加，WOE 也增加，WOE>0 表明这个区间内的响应客户占比更高。

IV 值是衡量变量对于分析目标提供的信息量大小的指标，简单来说，就是这个变量对目标变量的影响（相关性）有多大。一般来说，IV 值与相关性的对应关系如表 4-6 所示。

表 4-6 IV 值与相关性的对应关系（经验值）

IV	相关性
0.02-	无相关性
0.1-	弱相关性
0.3-	中等相关性
0.5-	强相关性
0.5+	可能计算错误

通常会选择 IV 值为 0.3 以上的变量进入模型，如果变量较少，也可降低对 IV 值的要求。

对于具有几百个以上变量的宽表，通过 IV 值筛选 50 个变量用于建模是常见的做法。

4.4.2 剔除共线性

建模的时候总是希望各个变量 x 与目标 y 有关系，x 之间没有关系，即 x 之间不存在共线性（有时会存在非线性关系），对于存在共线性的 x 变量进行剔除。通常采用方差膨胀因子或相关分析检验变量之间的共线性，通过主成分分析或因子分析生成新的一组变量，同时剔除共线性。这里介绍方差膨胀因子和主成分分析。

1. 方差膨胀因子

方差膨胀因子（Variance Inflation Factor，VIF）是最常用的多重共线性诊断指标，对于每个变量  $x_i$ ，其 VIF 的计算公式为： $VIF=1/(1-Rsquare)$ ，其中 Rsquare 是以  $x_i$  作为目标变量对其他变量 X 作回归分析的结果。Rsquare 是回归模型的重要评价指标，原理不多介绍，

只需记住一点: Rsquare 接近 1, 意味着  $x_i$  可以由其他变量  $X$  计算得到。与此一致, 某个变量  $x_i$  的 VIF 越大, 则它与其他变量之间的共线性越严重。

经验判断方法表明: 当  $0 < \text{VIF} < 10$  时, 存在较弱的多重共线性; 当  $10 \leq \text{VIF} < 100$  时, 存在较强的多重共线性; 当  $\text{VIF} \geq 100$  时, 存在严重的多重共线性, 一般要求 VIF 小于 5。

应用 SAS 中的 Reg 过程步可以计算变量的方差膨胀因子, 如代码清单 4-8 所示。运行一次过程步, 可以得到全部变量的 VIF, 而不需要每个变量分别运行一次。

代码清单 4-8

```
Proc Reg Data=Temp01;
```

```
Model Value=Age
```

```
Credit
```

```
Deposit
```

```
Dob
```

```
Ilt
```

```
Income
```

```
Loan
```

```
Rly_Fee_Amt
```

```
Rly_Loan_Amt
```

```
Rly_Loan_Cnt
```

```
R6m_Avg_Cdt
```

```
R6m_Avg_Dep
```

```
R6m_Avg_Ilt
```

```
R6m_Csh_Amt
```

```
R6m_Csh_Cnt
```

```
R6m_Loan_Amt
```

```
R6m_Txn_Amt
```

```
R6m_Txn_Cnt / VIF COLLIN COLLINOINT;
```

```
Run;
```

结果如表 4-7 所示, 最后一列就是各个变量的 VIF。Rly\_Loan\_Cnt 等四个变量的 VIF 特别大, 从上面介绍的内容可知, 它们之间存在明显的共线性。实际操作中会逐个剔除, 先把方差膨胀因子最大的变量剔除, 然后重新计算, 直到所有的方差膨胀因子都降到 5 以下。

## 2. 主成分分析

主成分分析本质上是一种降维算法, 当变量较多时, 有时会对变量进行信息加总, 可用少数几个综合变量代替原来较多的变量指标, 这几个综合变量能够尽可能多地反映原来变量的信息 (一般要求达 80% 以上), 彼此之间又相互独立, 不存在共线性。方法的原理可以参阅相关资料, 这里仅结合数据介绍实现过程。

表 4-8 为一组客户的信用卡行为数据, 共有 18 个变量, 具体含义不一一介绍。现在要做的是对这些变量进行主成分分析, 以去除共线性。代码清单 4-9 给出了主成分分析的 SAS 代码。



表 4-7 共线性检验运行结果

参数估计值						
变量	自由度	参数估计值	标准误差	t 值	Pr >  t	方差膨胀
Intercept	1	-29.93459	47.50313	-0.63	0.5286	0
AGE	1	1.71386	0.58267	2.94	0.0033	1.03983
CREDIT	B	-0.13379	0.25104	-0.53	0.5941	181325
DEPOSIT	1	0.01591	0.00086117	18.48	< 0.001	5.69511
DOB	1	0.21615	0.14696	1.47	0.1414	1.07262
ILT	1	0.00351	0.00285	1.23	0.2174	10.67163
INCOME	1	0.03264	0.02923	1.12	0.2641	1.00134
LOAN	B	0.33890	0.25103	1.35	0.1770	181312
R1Y_FEE_AMT	1	2.40193	2.54281	0.94	0.3449	17.67999
R1Y_LOAN_AMT	B	0.11632	1.37324	0.08	0.9325	4900574
R1Y_LOAN_CNT	0	0				
R6M_AVG_CDT	1	-0.01433	0.00125	-11.42	< 0.001	2.39403
R6M_AVG_DEP	1	0.00108	0.00086199	1.25	0.2103	5.96852
R6M_AVG_ILT	1	0.01264	0.00295	4.28	< 0.001	10.69572
R6M_CSH_AMT	1	0.00232	0.00007564	30.66	< 0.001	4.26105
R6M_CSH_CNT	1	0.34431	0.86445	0.40	0.6904	3.83125
R6M_LOAN_AMT	B	-0.23464	2.74654	-0.09	0.9319	4900750
R6M_TXN_AMT	1	0.00028942	0.00009394	3.08	0.0021	3.93222
R6M_TXN_CNT	1	1.28531	0.91480	1.41	0.1601	3.79164

表 4-8 信用卡客户行为数据 (temp01, 仅列出一部分作为示例)

Csr_ID	Value	Age	Income	Dob	Deposit	Credit	Ilit	Loan	R6m_Csh_Amt
56171	636	41	21	98	1287	1700	0	1703	110 073
61949	3008	41	54	173	5094	7366	0	7423	209 435
19544	234	48	864	164	700	0	1248	38	131
46381	2195	48	5	27	0	0	0	75	713 057
60148	3155	46	470	120	8008	5655	7	5662	426 314
26028	930	51	46	195	12 420	4505	0	4517	22 805
7663	215	59	16	52	3258	0	0	58	3978
20086	247	74	202	86	4314	0	0	62	1254
19094	452	33	1176	77	3841	1556	0	1653	2886
44149	1026	39	43	58	2078	1556	3483	1654	27 338
9342	345	36	42	45	1665	0	3	90	648
41386	314	30	180	116	1455	1164	0	1218	240
36341	1118	40	6	80	1386	2702	53	2705	8384

代码清单 4-9

```
Proc Princomp Data=Temp01 n=7 Out=Temp02;
```

```
Var Age
```

```
Credit
```

```
Deposit
```

```
Dob
```

```
Ilit
```

```
Income
```

```
Loan
```

```
R1y_Fee_Amt
```

```
R1y_Loan_Amt
```

```
R1y_Loan_Cnt
```

```

R6m_Avg_Cdt
R6m_Avg_Dep
R6m_Avg_Ilt
R6m_Csh_Amt
R6m_Csh_Cnt
R6m_Loan_Amt
R6m_Txn_Amt
R6m_Txn_Cnt;

```

```
Run;
```

\* 注释: n=7 表示反映全部变量 85% 信息量的主成分数, 即用 7 个综合变量替代原来的 18 个变量, 信息丢失 15%。

运行后将得到主成分累积贡献率, 用于告诉我们通过这些主成分能够反映多大比例的信息量, 如表 4-9 所示。

表 4-9 主成分分析的运行结果

相关矩阵的特征值				
特征值	差值	比例	累积	
1 4.49345894	1.38192268	0.2496	0.2496	
2 3.11153626	0.62213664	0.1729	0.4225	
3 2.48939962	0.59680380	0.1383	0.5608	
4 1.89259583	0.70048582	0.1051	0.6659	
5 1.19211001	0.12221698	0.0662	0.7322	
6 1.06989303	0.06472788	0.0594	0.7916	
7 1.00516515		0.0558	0.8475	

特征向量							
	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
AGE	-0.024918	-0.042185	-0.090075	0.115696	0.305958	0.506924	-0.079484
CREDIT	0.178921	0.339201	-0.337572	0.244044	-0.155263	-0.073475	-0.056096
DEPOSIT	0.060777	0.286149	0.369936	0.050497	0.387922	-0.353111	-0.068080
DOB	0.119637	-0.101834	-0.009393	-0.036008	0.224909	0.093480	-0.232643
ILT	0.083480	0.157895	0.473189	0.198411	-0.359823	0.261122	0.014400
INCOME	0.006668	-0.111793	-0.010800	-0.04222	-0.025656	-0.165539	0.917483
LOAN	0.178927	0.339200	-0.337557	0.244052	-0.155270	-0.073437	-0.056086
R1Y_FEE_AMT	0.424628	-0.201244	0.030707	0.074798	0.075689	0.009402	0.036218
R1Y_LOAN_AMT	0.431637	-0.203490	0.030120	0.074198	0.076904	0.007778	0.036154
R1Y_LOAN_CNT	0.431637	-0.203490	0.030120	0.074198	0.076904	0.007778	0.036154
R6M_AVG_CDT	0.183039	0.228727	-0.271041	0.239829	-0.172472	-0.144036	-0.060371
R6M_AVG_DEP	0.076999	0.361323	0.294849	0.061466	0.366646	-0.327209	-0.058157
R6M_AVG_ILT	0.085164	0.161504	0.471956	0.200189	-0.355740	0.262494	0.016943
R6M_CSH_AMT	0.130593	0.372311	-0.113315	-0.226038	0.185286	0.382572	0.154994
R6M_CSH_CNT	0.213577	0.074654	0.026319	-0.537036	-0.249370	-0.114745	-0.108590
R6M_LOAN_AMT	0.431636	-0.203494	0.030122	0.074193	0.076906	0.007785	0.036151
R6M_TXN_AMT	0.126256	0.357741	-0.040953	-0.272013	0.225955	0.361001	0.163380
R6M_TXN_CNT	0.214000	0.070273	0.031690	-0.535996	-0.255352	-0.128017	-0.107679

使用主成分降维后, 会得到数据表 temp02, 如表 4-10 所示, 可使用这 7 个主成分变量进行聚类等其他分析。

表 4-10 主成分分析后的数据 (temp02)

Csr_ID	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
56171	-0.61	2.98	3.10	0.81	3.48	-3.30	-0.64
61949	-2.02	2.99	2.39	0.53	2.62	-2.57	-0.61
19544	2.31	4.43	11.01	4.54	-3.62	2.11	-0.01

表 4-7 共线性检验运行结果

(续)

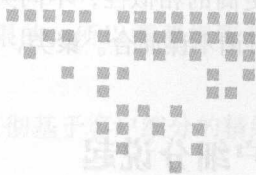
Csr_ID	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
43681	-0.58	2.04	2.26	0.71	3.26	-1.95	-0.96
60148	-0.93	2.20	2.21	0.68	2.48	-2.33	-0.43
26028	-2.36	2.87	2.16	0.22	1.68	-3.07	-0.30
7663	1.58	3.00	7.88	3.50	-2.30	1.85	0.61
20086	1.65	3.02	7.99	3.52	-2.44	1.86	-0.04
19094	0.52	0.76	1.53	0.70	1.62	-2.08	0.43
44149	-1.03	1.11	1.18	0.43	1.28	-1.41	-0.30
9342	1.57	1.71	5.74	2.66	-1.45	1.50	-0.28
41386	1.33	1.91	5.79	2.60	-2.00	0.89	0.66
36341	1.37	1.85	5.69	2.61	-1.83	1.04	0.33

观察累积贡献率，发现 7 个主成分的累积贡献率达到了 84.75%（代码中的 n=7 也是基于此确定的），满足了 85% 的最低要求，因此可以使用这 7 个主成分（综合变量）替代原来的全部变量进行其他的分析，这些变量间已不存在共线性问题。在很多教科书的主成分分析例子中，一般会出现 2 ~ 3 个主成分就能解释大部分变量的情况，但是实际的商业案例中很难达到这样的水平。需要说明的是，主成分是对原变量信息的加总，可解释性较差，可作为其他分析的数据预处理过程，但不建议将分析结果作为最终的结论。

4.5 本章小结

在要分析的原始数据中，存在大量缺失、重复、异常的数据，严重影响了数据挖掘算法的执行效率，并可能导致挖掘结果的偏差；为了获得更完整的分析结论，也需要我们基于原始数据设计衍生出更多的变量。为此，在开展数据挖掘算法之前，必须对原始数据进行预处理，而数据预处理花费的时间通常要占据整个分析流程 60% 的时间，充分了解了数据和要执行的分析算法，数据预处理才会事半功倍。

Csr_ID	Prin1	Prin2	Prin3	Prin4	Prin5	Prin6	Prin7
43681	-0.58	2.04	2.26	0.71	3.26	-1.95	-0.96
60148	-0.93	2.20	2.21	0.68	2.48	-2.33	-0.43
26028	-2.36	2.87	2.16	0.22	1.68	-3.07	-0.30
7663	1.58	3.00	7.88	3.50	-2.30	1.85	0.61
20086	1.65	3.02	7.99	3.52	-2.44	1.86	-0.04
19094	0.52	0.76	1.53	0.70	1.62	-2.08	0.43
44149	-1.03	1.11	1.18	0.43	1.28	-1.41	-0.30
9342	1.57	1.71	5.74	2.66	-1.45	1.50	-0.28
41386	1.33	1.91	5.79	2.60	-2.00	0.89	0.66
36341	1.37	1.85	5.69	2.61	-1.83	1.04	0.33



## 第5章 Chapter 5

## 聚类，简单易用的客户细分方法

物以类聚，人以群分。

——《周易·系辞上》

管理就是把复杂的问题简单化，混乱的事物规范化。

——杰克·韦尔奇

在产品同质化的市场环境下，企业之间的竞争已经由产品品质的竞争转变为顾客满意度的竞争，企业着眼于长远发展和领先市场的核心是针对不同人群的不同需求提供真正差异化的产品、服务以及营销策略。客户天生就存在差异，无论是行为特征、需求偏好还是价值贡献，因此同质化的营销策略在大量客户面前不仅毫无作用，还会让敏感的客户产生不被理解的感觉而选择离去。

企业如果想最大化地实现可持续发展，就需要专注正确的客户群体，找准客户的需求点，开展有针对性的营销。除了个别客户需要提供一对一的服务外，大部分情况下，企业都以细分客户群作为营销和服务单位。站在客户的角度，通过客户细分，企业可以识别不同需求的客户群，提供有针对性的产品和服务，满足不同客户群多样化的需求；站在企业的角度，客户细分能使企业将有限的资源优先投放到潜在的高价值客户，提高营销和服务的针对性，并能够对客户关系进行量化分析，为企业决策提供依据。

因此，客户细分是客户洞察、精准营销以及客户关系管理的基础，而在市场营销中，市场细分（market segmentation）、确定目标市场（market targeting）和市场定位（market position）也是其三大理论核心。聚类分析，就是将分析对象按照相似性划分归类，同一类



内的对象具有更高的相似性，不同类之间的对象存在差异，将共性和个性区分开来，与客户细分的理念可谓不谋而合。聚类，为了客户细分而生。

## 5.1 从客户细分说起

所谓客户细分，是指企业根据客户的属性、行为、需求、偏好以及价值等因素对客户进行分类。这里的客户也可以广义理解为我们所研究的对象，比如交易数据、产品信息以及病理信息等。以信用卡为例，客户注销卡是银行经常碰到的一个场景，面对这样的问题，银行该如何处理呢？是统一处理，还是每个客户分别对待？

信用卡公司会针对不同客户群的特征采取相应的反销卡措施，比如针对高端客户会通过赠予大量积分等方式极力挽留；针对一般客户会采取再交易即送额外多倍积分或者免年费等方式；而对于价值贡献较低甚至为负的客户，可以不做任何挽留。划分不同客户群的过程，就是客户细分。

### 5.1.1 为什么要做客户细分

生命都会有一个出生、成长、成熟、衰老、死亡的过程，对企业而言，客户也有一个类似的过程：潜在期、开发期、成长期、成熟期、衰退期和终止期，这便是客户生命周期，如图 5-1 所示。在每一个阶段，客户之间都存在特征差异，例如在成长期，客户的潜在价值不同；在成熟期，客户对产品的需求偏好不同；在衰退期，客户流失的倾向不同。通过细分可以识别出具有共性特征的客户群，单就营销来说，至少具有以下两个方面的意义。

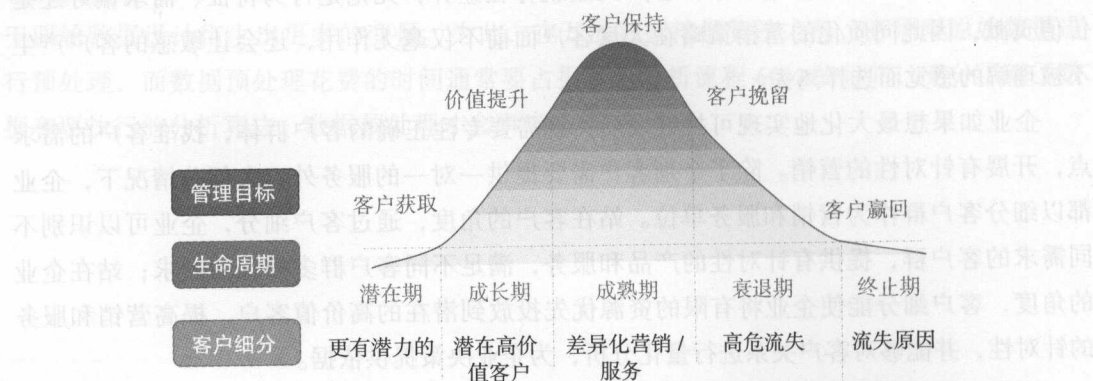


图 5-1 客户生命周期管理与客户细分

其一，有的放矢。客户细分后，可让我们对不同的客户群体有更深刻的认识，在此基

础上, 就可以针对性地提供不同的产品和服务, 达到事半功倍的效果。

其二, 资源优化。需求是无限的, 但是资源是有限的, 所以需要把资源投放到最需要的地方, 使其产生最大的效能。

因此, 在整个客户生命周期过程中, 企业都应该贯彻基于客户细分的精细化管理。

### 5.1.2 怎么做客户细分

通常来说, 可以按照业务场景以及关注的客户特征进行客户细分。常用的特征维度有五类: 利润贡献、生命周期、交易行为、需求偏好以及基于这些维度细分的多维组合, 如图 5-2 所示。

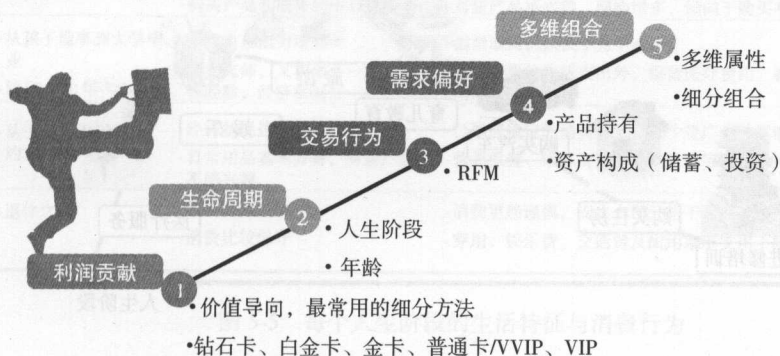


图 5-2 客户细分的常用特征维度

#### 1. 利润贡献

这是最常见的细分方法, 以客户产生的利润或者潜在价值进行细分, 比如银行通常将客户划分为钻石卡、白金卡、金卡、普通卡等等级。按照客户价值金字塔 (见图 5-3), 高价值的客户占比非常小, 但其利润贡献占比却非常大, 相关统计表明, 只有大约 20% 的客户能给银行带来收益, 因此找到这 20% 的优质客户就成为银行的一大目标。

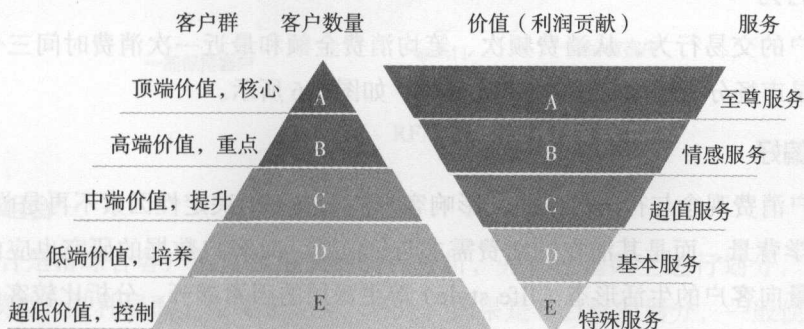


图 5-3 客户价值金字塔

## 2. 生命周期

以客户生命周期 (life cycle) 为主线, 根据客户信息以及长期的交易记录, 寻找生活形态有明显差异的时点, 整理出各阶段的消费特征, 分为 6 个人生阶段, 如图 5-4 所示。这里的客户生命周期是指围绕家庭组建与解体的人生过程, 有别于以产品为导向的客户生命周期。

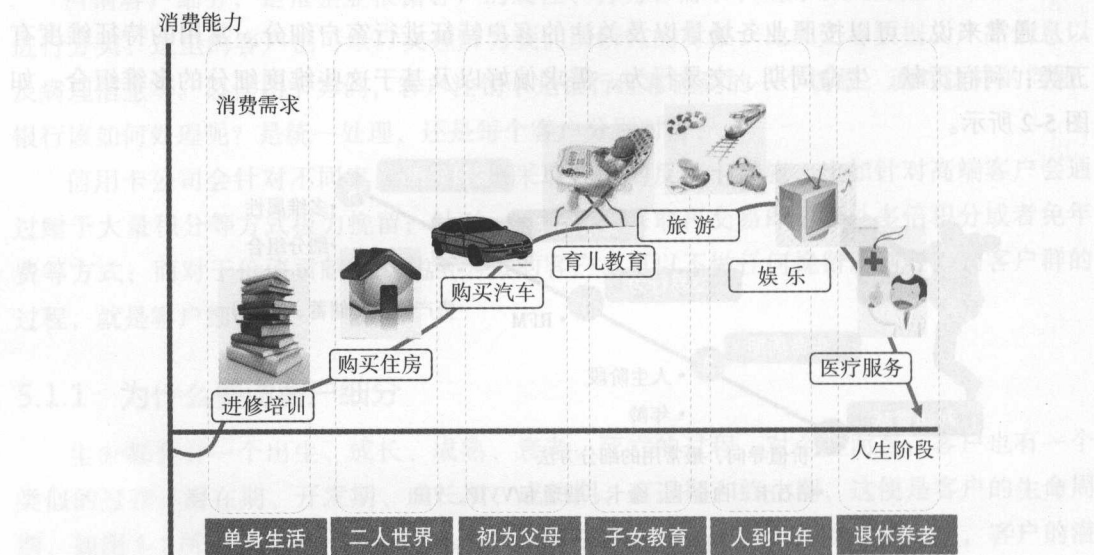


图 5-4 客户人生阶段的划分

针对每个人生阶段, 可详细分析客户的生活特征和消费行为, 如图 5-5 所示。客户生命周期由家庭发展的各个阶段构成, 考虑了年龄、婚姻状况、家庭结构等因素。每一个阶段的客户都有着许多共同的消费观念、购买需求和取向, 不同阶段的客户有明显的差异。客户生命周期的划分与标识, 对于细分市场和制定营销策略来说是非常有用的。

## 3. 交易行为

基于客户的交易行为, 从消费频次、笔均消费金额和最近一次消费时间三个维度进行细分, 这便是市场分析领域常用的 RFM 模型, 如图 5-6 所示。

## 4. 需求偏好

随着客户消费观念与行为的成熟, 影响客户消费决策的决定性因素不再是收入、年龄、职业等统计学背景, 而是其潜在的消费需求与价值观。对客户数据的研究也应由一般性的社会统计变量向客户的生活形态 (life style) 等更深层的因素展开。分析比较客户的消费习惯、消费场所、生活方式等各个方面的异同, 整理出一些有特定生活形态的客户群, 为客

户打上标签。此处结合信用卡的消费特点归纳了 10 类客户群, 如图 5-7 所示, 每类客户的认定方法以及可能的营销与服务机会如图 5-8 所示。

人生阶段	周期	生活特征	消费行为/需求
单身生活	·从跨入职场到恋爱前 ·约1~5年	·几乎没有经济负担 ·消费旺盛	·消费观念紧跟潮流 ·注重娱乐产品和基本生活必需品的消费
二人世界	·从恋爱到结婚, 孩子尚未出生 ·约1~5年	·经济收入增加且生活稳定 ·具有比较大的需求量和比较强的购买力	·有充足的时间旅游和娱乐 ·要准备结婚用品, 如住宅、家具、服装、结婚仪式、蜜月旅行等 ·耐用消费品的购买量较高, 往往需要购买住房
初为父母	·孩子刚出生阶段 ·一般为3年	·生活方式较以前有了很大变化, 支出增多, 消费力下降 ·购买产品和服务的中心是孩子	·消费趋向理性, 会购买大量的生活必需品, 从食品、服装、玩具、药品到文化学习用品 ·对新产品感兴趣, 网购增多, 倾向于购买有广告的产品
子女教育	·从孩子懂事到大学毕业 ·约15~20年	·子女自理能力增强 ·精力充沛, 又积累了一定的工作经验, 经济状况好	·消费取向仍以孩子为中心 ·除了必要的生活支出外, 保健医疗费用、教育服务费用、智力开发费用增多
人到中年	·从子女工作到退休, 约15年	·经济状况达到高峰状态 ·日常用品基本齐备, 对新产品不感兴趣	·已形成稳定的消费习惯, 极少受广告的影响 ·营养保健、锻炼身体、外出旅游等增多, 可能购买娱乐品和奢侈品
退休养老	·退休之后	·收入大幅度减少 ·消费比较保守	·消费更趋谨慎, 支出大部分用于食品、医疗保健和社会服务方面 ·穿用、娱乐费、交通费及耐用家电支出下降

图 5-5 每个人生阶段的生活特征与消费行为

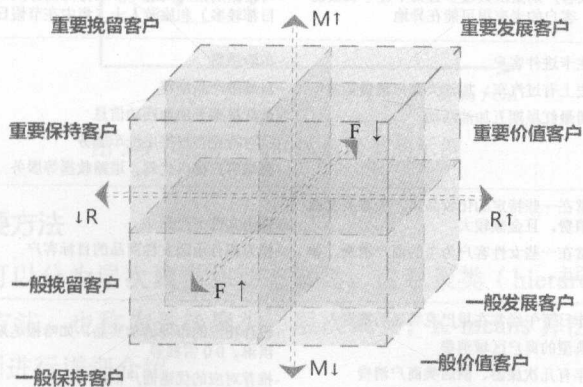


图 5-6 RFM 模型

## 5. 多维组合

多维组合是指综合客户的多维度属性进行分析, 并在此基础上进行划分, 比如之前提到的 RFM 模型, 某种层面上便属于此类。另外, 如果是有目标的划分, 一般使用决策树模型; 如果是无目标的划分, 一般使用聚类分析。





图 5-7 特定生活形态客户群

人生阶段	认定方法	营销与服务机会
公司白领	<ul style="list-style-type: none"> <li>这类客户经常在外吃工作餐，中午经常有餐饮类消费，往往是2~4人的消费额，如50~200元</li> <li>中午与下午经常在一些公司集中的商圈内消费，工作时间段基本无消费</li> </ul>	<ul style="list-style-type: none"> <li>推荐周边餐馆的优惠信息</li> <li>联合优惠商户专为我行客户推出午餐套餐</li> <li>以刷卡金的形式设定午餐基金</li> </ul>
交际应酬	<ul style="list-style-type: none"> <li>晚间消费多，金额较大</li> <li>消费场所多为较高价位的餐饮与KTV、浴场等娱乐场所，且往往兼具两类</li> <li>消费地域分散、异地消费较多</li> </ul>	<ul style="list-style-type: none"> <li>对于高端客户提供酒后代驾服务，尽可能分析客户的行为规律，注重接触客户的时间</li> </ul>
商旅人士	<ul style="list-style-type: none"> <li>外籍人士，或客户身份证上的出生地不同于发卡行或当前消费地</li> <li>有较多的购票记录</li> <li>如果客户购票次数较少且集中在小长假前后，客户的老家很可能在异地</li> </ul>	<ul style="list-style-type: none"> <li>旅游产品推荐，可作为积分乐园商旅频道的目标客户</li> <li>节假日特别是春节期间提供便民服务，例如购票、包机等</li> <li>可根据购票时间进一步细分为商务人士（平时与节假日都较多）和旅游人士（集中在节假日和周末）</li> </ul>
开车一族	<ul style="list-style-type: none"> <li>车主卡进件客户</li> <li>历史上有过汽车、加油类商户消费记录</li> <li>注册最红星期五加油活动</li> </ul>	<ul style="list-style-type: none"> <li>车险销售</li> <li>自驾游产品推荐</li> <li>最红星期五加油活动信息</li> <li>针对特定客户推出洗车服务</li> <li>高端客户提供代驾、道路救援等服务</li> </ul>
时尚女性	<ul style="list-style-type: none"> <li>经常在一些特定的化妆品店、美容美发商户消费，且金额较大</li> <li>经常在一些女性客户为主的商户消费，如屈臣氏</li> </ul>	<ul style="list-style-type: none"> <li>推出女性卡产品</li> <li>作为积分乐园女性商品的目标客户</li> </ul>
小资阶层	<ul style="list-style-type: none"> <li>工作日下午经常在星巴克等场所消费</li> <li>在典型的商户区域消费</li> <li>每年有几次旅游、酒店类商户消费</li> </ul>	<ul style="list-style-type: none"> <li>设计相应的市场活动奖品，如哈根达斯冰淇淋、DQ雪糕等</li> <li>推荐对应的优惠商户信息</li> <li>发卡的目标区域，指导发卡</li> </ul>
奢侈品达人	<ul style="list-style-type: none"> <li>有知名品牌（如LV）、高端商户（如上海的恒隆百货）的消费记录</li> <li>有一定数量的大额消费</li> </ul>	<ul style="list-style-type: none"> <li>推荐高端优惠商户</li> <li>积分乐园奢侈品的目标客户群</li> </ul>
运动健将	<ul style="list-style-type: none"> <li>近期在健身场所有过消费（如一兆韦德、舒适堡等）且金额较大</li> </ul>	<ul style="list-style-type: none"> <li>联名卡经营</li> <li>积分乐园运动户外商品销售</li> </ul>
果粉	<ul style="list-style-type: none"> <li>在苹果体验店与专营店有较多消费</li> </ul>	<ul style="list-style-type: none"> <li>积分乐园苹果产品的销售</li> <li>奖品设置</li> </ul>

图 5-8 不同生活形态的客户群认定及可能的营销与服务机会

### 5.1.3 聚类分析, 无监督的客户细分方法

在市场研究领域, 客户细分特别需要业务领域的经验和统计学知识, 因此分类会带有一定的主观性(中性词), 对于菜鸟级分析人员来说, 具有较高的挑战性。聚类, 是数据挖掘领域最基本的方法, 可基于多维属性进行客观的划分归类, 操作人员只需要按照格式准备好数据并设定几个参数就可轻松搞定, 在客户细分领域得到了广泛应用。

#### 1. 聚类分析的基本思想

所谓“物以类聚, 人以群分”, 简单来说, 聚类分析就是根据对象之间的相似度, 在没有任何先验知识的情况下进行分类。聚类分析是一种无监督学习, 遵循的原则是在分类后, 应尽可能地保证同类之间具有较高的相似性, 而不同类之间具有较低的相似性。

举个简单的例子, 假如根据身高和体重对某班男生进行分组, 可以把每个男生想象为二维空间中的一个点, 然后距离近的合并成一类, 如图 5-9 所示。

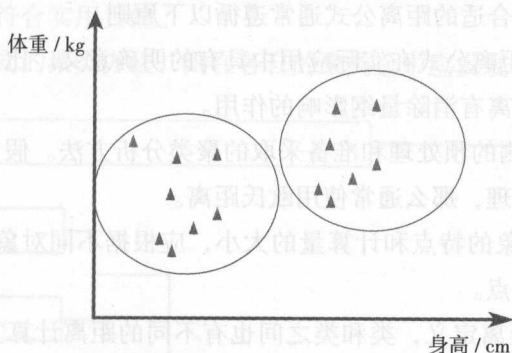


图 5-9 根据身高和体重聚类的示例

#### 2. 聚类分析的主要方法

聚类分析按方法可以分为层次聚类和动态聚类。谱系聚类(hierarchical clustering)是层次聚类中最典型的方法, 也称为系统聚类、层次聚类; K-means 算法是动态聚类中最经典的方法。下面将分别进行详细介绍。

## 5.2 谱系聚类

### 5.2.1 基本步骤

谱系聚类采用自下而上的汇总过程, 简单来说, 就是每次把距离最近的观测合并起来,

直到最终合并为一类，具体步骤如下。

- 1)  $N$  个观测各自成一类。
- 2) 计算  $N$  个观测两两之间的距离，得到一个距离矩阵。
- 3) 合并距离最近的两个观测为一类。
- 4) 重复上述过程，直到所有观测合并为一类。
- 5) 画谱系。
- 6) 决定类的个数。

其中，需要特别关注两个问题，距离的选择和聚类数的确定。

### 1. 距离的选择

观测之间可以使用不同的方法来定义距离（相似度），常见的有欧式距离、明氏距离、马氏距离、兰氏距离等，不同的距离公式侧重点和实际意义不同，因此最终的聚类结果也会有所不同，感兴趣的读者可以查阅这几个距离的计算公式，以观察它们的区别。总体来说，在聚类分析时，选择合适的距离公式通常遵循以下原则。

- 1) 需要考虑所选择距离公式在实际应用中具有明确意义。比如欧式距离就有很明确的空间距离概念；马氏距离有消除量纲影响的作用。
- 2) 需要综合考虑观测的预处理和准备采取的聚类分析方法。假如在聚类分析之前，已经对变量进行了标准化处理，那么通常使用欧氏距离。
- 3) 需要考虑研究对象的特点和计算量的大小，应根据不同对象的特点做出具体分析，并考虑不同距离公式的缺点。

观测之间有不同的距离定义，类和类之间也有不同的距离计算方法，由此也产生了不同的谱系聚类方法，下面介绍五种常用的方法。

- 1) 最短距离法：用于定义类与类之间距离为两类中最近观测的距离，使空间浓缩，形成链状，但是分类效果可能会不好。
- 2) 最长距离法：用于定义类与类之间距离为两类最远观测的距离，受异常值的影响比较大。
- 3) 重心法：以两类重心之间的距离作为两类间的距离。重心即该类观测的均值。每合并一次类，都要重新计算新类的重心，因此在处理异常值方面比较稳健，但是未充分利用观测的信息。
- 4) 类平均法：以两类观测两两之间距离平方的平均作为类间距离的平方。
- 5) 离差平方和法：又称 Ward 法，其基本思想是认为同类样品的离差平方和应当较小，类与类的离差平方和应当较大。首先  $n$  个样品各自成一类，然后每次缩小一类，每缩小一类离差平方和就要增大，选择使离差平方和增加最小的两类合并，直到所有样品归为一类。

这类方法分类效果较好，应用较广泛，对异常值较敏感。

与观测间距离公式类似，采用不同的类间距离得到的结果也会不同，至今还没有合适的衡量标准，因为不存在一种最优方法。实际操作中，应结合分析对象的特点并使用多种方法提炼共性。

## 2. 聚类数的确定

谱系聚类最终可以得到一棵树，使所有观测都聚成一类，但是到底应该分几类，并没有标准，一般通过聚类谱系图（又称龙骨图）和 R 方统计量来确定。

图 5-10 是聚类分析教科书中常见的一张谱系图，从中可以较为清楚地分为三类，其中北京和上海为一类，河北、河南、山西和内蒙古为一类，剩余为第三类。使用谱系图确认聚类数时，需要注意以下问题。

- 1) 各类重心的距离必须很大。
- 2) 确定的类中，各类所包含的元素都不要太多。
- 3) 类的个数必须符合实用目的。
- 4) 若采用几种不同的聚类方法，则在各自的聚类途中应发现相同的类。

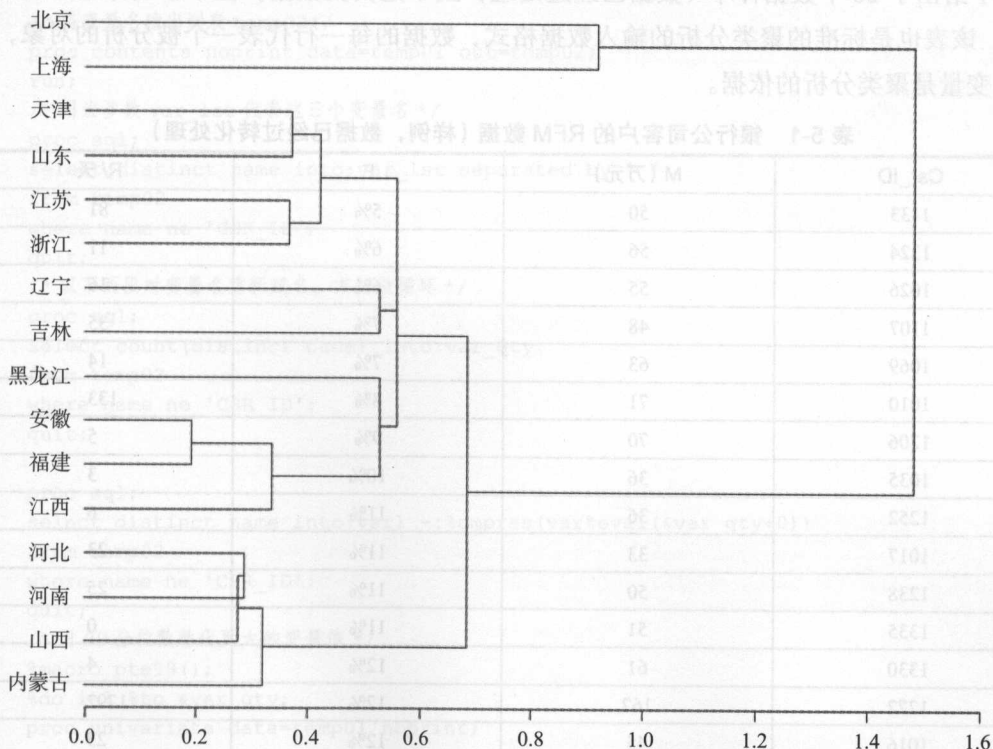


图 5-10 聚类谱系图 (来源: 百度图片)



R 方统计量,是指类间的离差平方和占有所有离差平方和的比例,R 方越大,说明每个类内的离差平方和越小。简单来说,R 方越大,分类越合适。是否意味着 R 方最大的分类就是最优解?不是,因为 R 方会随着分类数量的增加而单向变大。通常的做法是,画出不同分类数的 R 方折线图,以拐点对应的分类数量作为最佳划分。

## 5.2.2 案例:公司客户差异化服务

对公业务是商业银行最传统的业务,主要有存款业务、贷款业务、项目贷款、结算业务、国际业务(结售汇、信用证、保函、押汇)等。随着零售业务和中间业务的发展,对公业务带来的利润占比逐步下降,但在大部分商业银行中仍占据六成以上的比例。可以说,对公业务是银行生存的根本,维护好公司客户关系至关重要。某商业银行希望对现有的公司客户进行细分,针对各类客户群的特征提供差异化的产品和服务,以提高客户满意度。

在业务划分基础上,针对 336 个项目贷款客户,应用 RFM 数据进行进一步分类。R 是客户最近一次有效交易(为银行带来利润的交易)距离今天的天数,F 是客户总体活跃度指数(包括交易、查询、账户管理等,全部公司客户统一折算),M 是客户带来的累计利润,表 5-1 给出了 20 个数据样本(数据已经过处理,虽不是真实数据,但不影响介绍方法的需要)。该表也是标准的聚类分析的输入数据格式,数据的每一行代表一个被分析的对象,每一个变量是聚类分析的依据。

表 5-1 银行公司客户的 RFM 数据(样例,数据已经过转化处理)

Csr_ID	M (万元)	F	R/天
1133	50	5%	81
1324	56	6%	11
1026	55	6%	16
1107	48	7%	35
1069	63	7%	14
1010	71	8%	133
1206	70	9%	5
1035	36	10%	3
1252	36	11%	6
1017	33	11%	23
1238	50	11%	25
1335	51	11%	0
1330	61	12%	4
1272	162	12%	1292
1016	45	12%	23
1270	60	13%	140

(续)

Csr_ID	M (万元)	F	R/ 天
1201	91	13%	6
1321	40	14%	64
1193	68	14%	18
1124	45	14%	52

聚类分析本身并不复杂, 但一个完整的聚类过程至少应该包括异常值处理、相关分析、降维分析(如变量间存在相关性)、数据标准化和聚类, 并且应根据聚类指标评估进行迭代。

### 1. 异常值处理

聚类算法对异常值或特别大的极值非常敏感, 需要对每一个变量的数据质量进行检查, 以排除异常值的影响。对于真实存在的极大值, 要视分析的实际需要做处理, 分位数替代是常用的方法。例如, 对于 99 分位点以上的数据, 用 99 分位点代替。如果变量数值波动大, 还可以对变量做取对数的平滑处理。代码清单 5-1 是 99 分位数替代的 SAS 代码。

代码清单 5-1

```

/* 将变量名输出到表 temp02 */
proc contents noprint data=temp01 out=temp02;
run;
/* 用宏参数 var_lst 代表这三个变量名 */
proc sql;
select distinct name into:var_lst separated by ' '
from temp02
where name ne 'CSR_ID';
quit;
/* 以下两段对变量名重新命名, 方便做循环 */
proc sql;
select count(distinct name) into:var_qty
from temp02
where name ne 'CSR_ID';
quit;

proc sql;
select distinct name into:var1 -:%cmpres(var%eval(&var_qty+0))
from temp02
where name ne 'CSR_ID';
quit;
/* 用 99 分位数替代更大的变量值 */
%macro pts99();
%do i=1 %to &var_qty;
proc univariate data=temp01 noprint;
var &&var&i;
output out=tmp_&i p99=&&var&i.._p99;
run;

```

```

data temp03;
  set temp01;
  if (_n_=1) then set tmp_&i;
  if &&var&i>&&var&i.._p99 then &&var&i=&&var&i.._p99;
  drop &&var&i.._p99;
run;
%end;
%mend;
%pts99;

```

## 2. 相关性检验

SAS 中可以使用 CORR 过程步计算变量间的相关性 (见代码清单 5-2)。对于存在强相关性的变量, 可以剔除其中一个, 或者使用主成分分析做降维, 同时剔除相关性。相关分析结果表明, 变量间无明显相关性, 且变量均具有明确的业务含义, 所以此处无需做降维处理。

代码清单 5-2

```

proc corr data=temp03 out=temp5;
var &var_lst.;
run;

```

## 3. 数据标准化

标准化的目的是去除量纲, 将数值分布在不同值域的各个变量转换到同一个值域内, 每个变量在计算距离的时候被同等对待。代码清单 5-3 是将数据标准化均值为 0、标准差为 1 的正态分布。

代码清单 5-3

```

proc standard data=temp04 out=temp05 mean=0 std=1;
var &var_lst.;
run;

```

## 4. 谱系聚类过程

经过这些预处理步骤, 终于可以运行聚类算法了。代码清单 5-4 是 SAS 里提供谱系聚类的过程步。

代码清单 5-4

```

proc cluster data=temp04 outtree=tree method=ward;
var &var_lst.;
copy csr_id;

```

```
run;

proc tree data=tree nclusters=4 out=output ;
  copy csr_id &var_lst.;
run;
```

这里使用的是 cluster 过程步, 并使用 ward 法做聚类, 其中 method 选项中共有 11 种谱系聚类的算法可供选择, 具体可查询 SAS 帮助。另外, 在 cluster 过程步中, 可以设定 standard 参数进行数据标准化, 因此之前就不需要单独进行数据标准化了。

运行完以上代码, SAS 将输出聚类谱系图 (见图 5-11) 和 R 方统计量 (见图 5-12), 并据此画出折线图。结合两个图, 划分为四类比较合适。从聚类谱系图也可看出, 当聚类对象较多 (比如超过 1000 个) 时, 这个图是没法看的, 因此谱系聚类更适合轻量级的聚类场景。

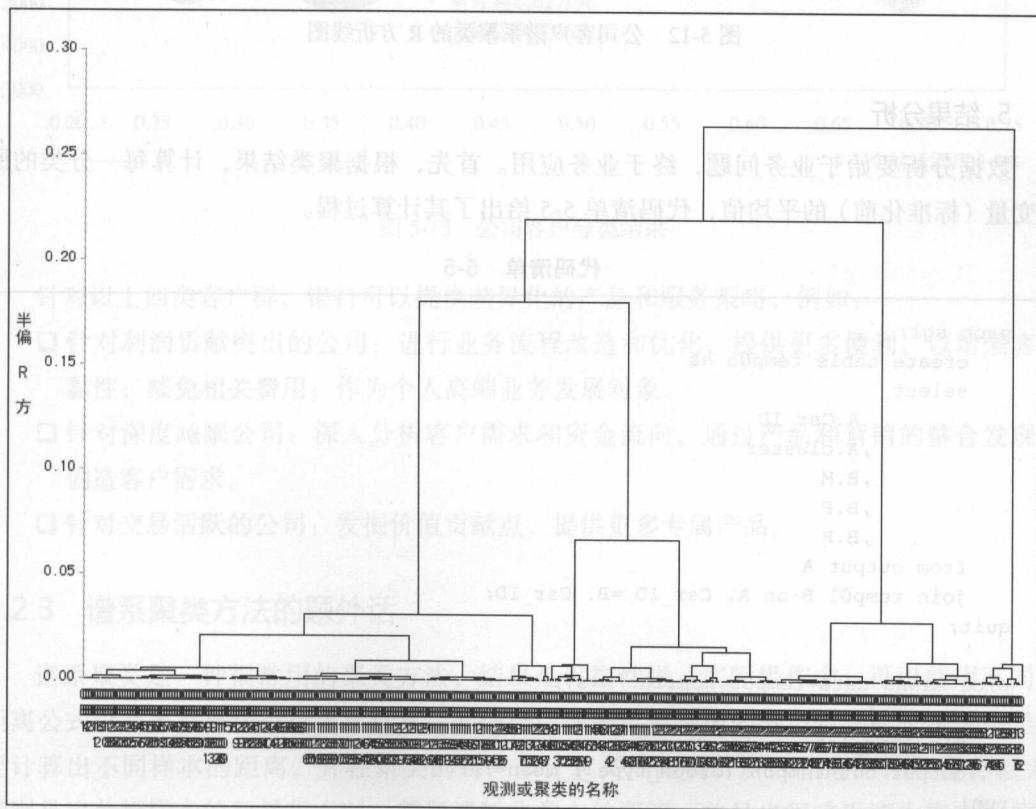


图 5-11 公司客户的聚类谱系图



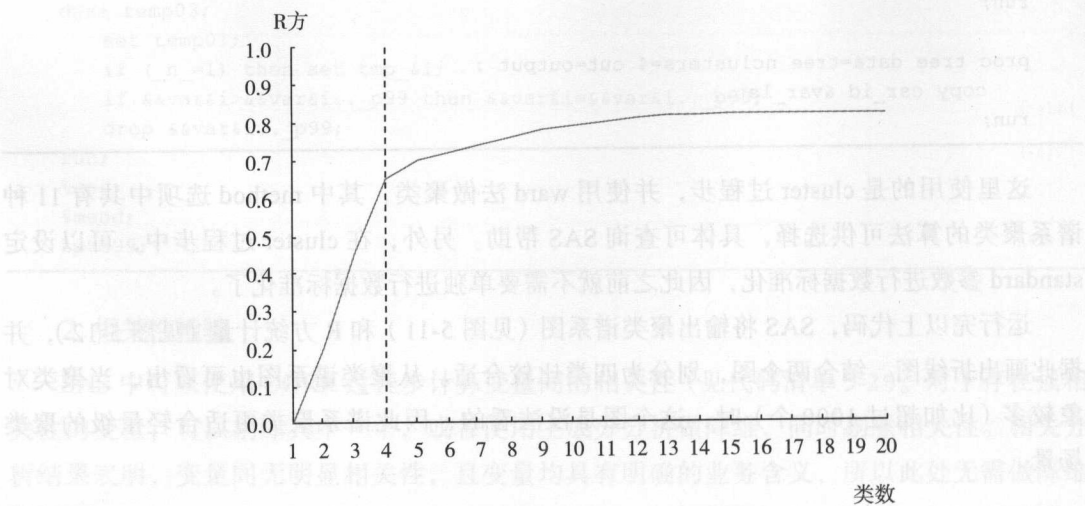


图 5-12 公司客户谱系聚类的 R 方折线图

5. 结果分析

数据分析要始于业务问题，终于业务应用。首先，根据聚类结果，计算每一分类的原始变量（标准化前）的平均值，代码清单 5-5 给出了其计算过程。

代码清单 5-5

```
proc sql;
  create table temp05 as
  select
    A.Csr_ID
  ,A.cluster
  ,B.M
  ,B.F
  ,B.R
  from output A
  join temp01 B on A. Csr_ID =B. Csr_ID;
quit;

proc means data=temp05 noprint nway;
  class cluster;
  var &var_lst;
  output out=temp06 (drop=_type_) mean=;
run;
```

图 5-13 展示了四类公司客户，并根据主要特征作了解读，便于业务人员理解。

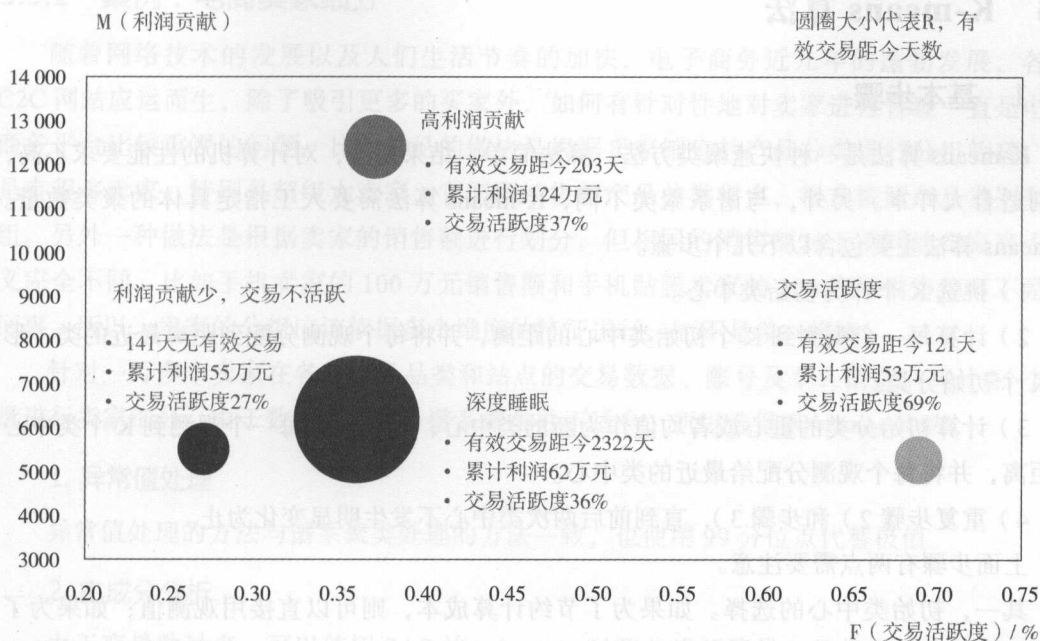


图 5-13 公司客户分类结果

针对以上四类客户群, 银行可以提供差异化的产品和服务策略, 例如:

- 针对利润贡献突出的公司: 进行业务流程改造和优化, 提供更多便利, 以增强客户黏性; 减免相关费用; 作为个人高端业务发展对象。
- 针对深度睡眠公司: 深入分析客户需求和资金流向, 通过产品和营销的整合发现或创造客户需求。
- 针对交易活跃的公司: 发掘价值贡献点, 提供更多专属产品。

### 5.2.3 谱系聚类方法的题外话

谱系聚类是一种很常用的聚类方法, 结果是可视性强, 实际操作中, 可以使用不同的距离公式以及聚类算法进行多次计算, 并结合业务确定最理想的分类数。由于谱系聚类需要计算出不同样本的距离, 并在聚类的每一步计算类间距离, 所以相应的计算量比较大, 特别是当分析样本的数量很大时, 需要消耗非常大的资源, 并且也很难看清大样本下的聚类谱系图, 这给应用带来了一定的困难, 因此谱系聚类更适合小样本的聚类场景。

## 5.3 K-means 算法

### 5.3.1 基本步骤

K-means 算法是一种快速聚类方法, 算法简单, 结果易懂, 对计算机的性能要求不高, 特别适合大样本。另外, 与谱系聚类不同, K-means 算法需要人工指定具体的聚类数量。K-means 算法主要包含以下几个步骤。

- 1) 挑选 K 个作为初始类中心。
- 2) 计算每一个观测到 K 个初始类中心的距离, 并将每个观测分配到距离最近的类, 形成 K 个初始分类。
- 3) 计算初始分类的重心或者均值作为新的类中心, 重新计算每一个观测到 K 个类中心的距离, 并将每个观测分配给最近的类中心。
- 4) 重复步骤 2) 和步骤 3), 直到前后两次类中心不发生明显变化为止。

上面步骤有两点需要注意。

其一, 初始类中心的选择。如果为了节约计算成本, 则可以直接用观测值; 如果为了提高精度, 则可以使用合成法, 这里使用的是一种叫 Random Centroids 的合成法。其基本思想是, 先形成一个分类数较大的初步分类, 比如 100, 然后使用这 100 个分类的质心作为进一步聚类的备选初始类中心。由于初步分类的个数较大, 不同的类可以很好地分开, 因此将其质心作为下一步分类的初始点可以提高聚类效果。需要注意的是, 由于初始点的不同, 每次的结果也会有所不同, 所以建议多运行几次, 选择最优结果。

其二, K 值的确定。除非有明确的业务逻辑, K 值在聚类分析前是不确定的, 一般结合业务经验和统计指标共同确定。具体做法如下(会出现很多英文字符, 具体含义可参看后面的案例)。

- 1) K 值取 2 ~ N, 对每个 K 使用不同的初始点跑 M 次。
- 2) 使用每个 K 中最小的 SSE (误差平方和) 作为这个 K 的 SSE。
- 3) 使用 K 和 SSE 作图, 拐点处即为最终确定的 K 的取值。
- 4) 所确定的 K 中, M 次运行中 SSE 最小的那次, 即为最优结果。

此外, 还有以下三个评价聚类质量的指标。

□ ERSQ= 近似期望总体 R 方: 模型总体拟合度, 评价类间差异是否够大, 理想为 1。

□ CCC= 立方聚类准则: 评价类内相似性是否够大, 一般要求大于 2; 如果为较大的负数, 很可能变量存在异常值。

□ PSF= 伪 F 统计量: 评价聚类数量是否合理, 越大越好。

### 5.3.2 案例：电商卖家细分

随着网络技术的发展以及人们生活节奏的加快, 电子商务近几年的蓬勃发展, 各种 C2C 网站应运而生, 除了吸引更多的买家外, 如何有针对性地对卖家进行管理一直是电子商务平台比较重视的问题。比较常见的做法是根据卖家销售的产品分类进行分组管理, 但是有很多卖家, 特别是超级大卖家, 它们所销售的产品分类很多, 很难按照产品类别来分组。另外一种做法是根据卖家的销售额进行划分, 但相同的销售额对于不同的卖家来说意义完全不同, 比如手机卖家的 100 万元销售额和手机贴膜卖家的 100 万元销售额可不是一回事。所以, 卖家的分组应该依据多个维度的特征进行, 而不是单一维度。

针对一万多个卖家在各个产品品类和站点的交易数据、账号及个人信息, 共计 25 个变量进行卖家细分。由于数据量较大, 谱系聚类并不适合, 所以选择 K-means 算法。

#### 1. 异常值处理

异常值处理的方法与谱系聚类处理的方法一致, 也使用 99 分位点代替极值。

#### 2. 主成分分析

由于变量数过多, 可以使用 SAS 的 princomp 过程步进行降维, 具体方法参阅第 4 章。最终, 10 个主成分的累积贡献率达到 89% 以上, 如表 5-2 所示, 用于替代原来的 25 个变量进行聚类。

表 5-2 主成分分析结果

相关矩阵的特征值				
	特征值	差值	比例	累积
1	4.53843276	1.50948452	0.1891	0.1891
2	3.02894824	0.24844518	0.1262	0.3153
3	2.78050307	0.53312446	0.1159	0.4312
4	2.24737860	0.44228399	0.0936	0.5248
5	1.80509461	0.20162989	0.0752	0.6000
6	1.60346473	0.02492396	0.0668	0.6668
7	1.57854076	0.11066641	0.0658	0.7326
8	1.46787436	0.15676332	0.0612	0.7938
9	1.31111104	0.25323091	0.0546	0.8484
10	1.05788013		0.0441	0.8925

#### 3. 数据标准化

同 5.2.2 节相关内容, 这里不再赘述。



#### 4. 转换成球面数据

K-means 算法可以通过设定类中心和欧式距离进行分类, 尤其适合球形分布的数据, 如果是细长型分布的数据或者非凸型数据, 算法的表现就会相对较差。一种很自然的变通想法就是, 将非球形数据变换成球形数据。代码清单 5-6 给出的 SAS 代码将主成分转换成球面数据。

代码清单 5-6

---

```
proc aceclus data=temp01 proportion=.2 maxiter=25 out=temp02;
var prin;;
run;
```

---

#### 5. 聚类循环迭代

K-means 属于有监督算法, 需要人工指定聚类数量  $K$ , 但究竟  $K$  应该取多少, 很难主观决定。一种解决方法是,  $K$  取 2 ~ 10 (考虑到业务策略, 聚类数量不建议太多), 每个  $K$  运行 10 次, 根据 SSE 的结果来判断  $K$  的取值。代码清单 5-7 是 K-means 循环迭代的 SAS 过程代码。

代码清单 5-7

---

```
data clus_stat;
sse=.;
pseudo_f=.;
ersq=.;
ccc=.;
test_n=.;
cluster_n=.;
stop;
run;

%macro random_centriods(dataset=,test_n=,cluster_n=,iter=);
%do i=1 %to &test_n;
proc fastclus data=&dataset maxc=100 maxiter=0 outseed=seed replace=
random random=&i noprint;
var Can;;
run;

proc surveyselect data=seed seed=&i n=&cluster_n method=srs out=in_seed noprint;
run;

%do k=2 %to &cluster_n;
data cluster_seed;
set in_seed(obs=&k);
run;
```

---

```

proc fastclus data=&dataset seed=cluster_seed maxc=&k maxiter=&iter converge=
0 out=out_&i._&k outstat=outstat_&i._&k;
var Can;;
run;

data sse;
set out_&i._&k end=eof;
x=distance*distance;
sse+x;
if eof=1 then output;
keep sse;
run;

data ccc;
set outstat_&i._&k;
if _type_ in ('CCC','ERSQ','PSEUDO_F');
keep _type_ over_all;
run;

proc transpose data=ccc out=ccct(drop= _name_ _label_);
id _type_;
run;

data temp;
merge sse ccct;
test_n=&i;
cluster_n=&k;
run;

proc append base=clus_stat data=temp;
run;
%end;
%mend random_centriods;
%random_centriods(dataset=temp02,test_n=10,cluster_n=10,iter=99);

proc print data=clus_stat noobs;
run;

```

数据集 clus\_stat 保存了每次运行的结果, 表 5-3 为其中的一部分, 包含 K 值、运行次数以及 SSE、PSF、ERSQ、CCC 等评价指标。

以上数据经过简单处理后, 画出 SSE 折线图, 如图 5-14 所示, 聚成 8 类比较合理。相关的分类信息以及具体每条观测的分类结果分别保存在表 Outstat\_8\_8 以及 Out\_8\_8 中。

通过数据投影技术, 将 10 个主成分聚类得到的 8 组卖家展示到二维图上, 如图 5-15 所示, 各组卖家数较为均匀 (对卖家作了有效划分), 分布相对分散 (体现出了卖家差异), 接下来可针对每组卖家设定差异化管理策略。数据投影方法有多种, 可参阅关于矩阵分解的相关材料。

表 5-3 K-means 算法循环迭代的运行结果

sse	pseudo_f	ersq	ccc	test_n	cluster_n
584881.94		0.00000	0.0000	1	1
160555.31	26862.12	0.62777	37.8881	1	2
124008.16	18885.29	0.74405	19.1089	1	3
116487.41	13620.43	0.78477	7.3874	1	4
109982.15	10968.73	0.80363	4.0108	1	5
104480.43	9343.15	0.81389	3.7367	1	6
99737.58	8235.92	0.82054	6.5207	1	7
95162.12	7467.82	0.82557	12.4689	1	8
90167.60	6965.94	0.82974	23.4086	1	9
78560.82	7272.78	0.83334	57.3509	1	10
584881.94		0.00000	0.0000	2	1
160555.31	26862.12	0.62777	37.8881	2	2
153139.04	14326.21	0.74405	-2.3060	2	3

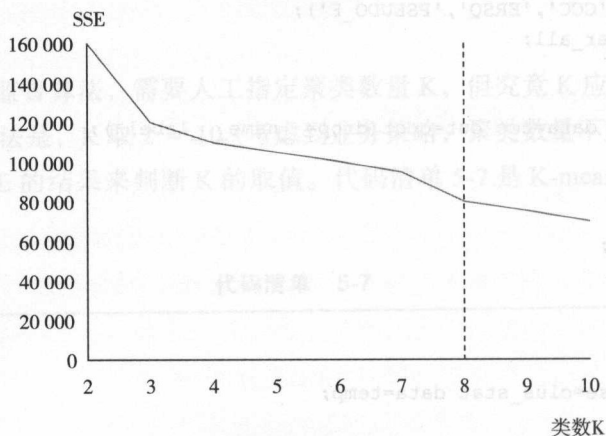


图 5-14 聚类数量 K 对应的 SSE

第一维度

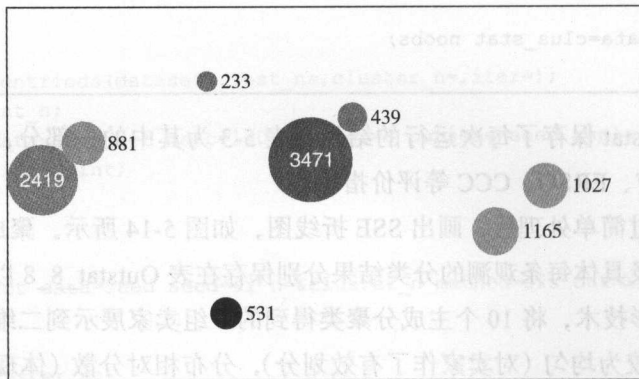


图 5-15 卖家聚类结果

### 5.3.3 K-means 算法的题外话

K-means 算法能够针对多维数据进行聚类, 算法简单易行, 结果直观易懂。该算法仅适用于连续型变量, 不同的 K 值和初始点得到的聚类结果存在差异。上面提到的聚类循环迭代方法能够帮助获得相对较优的结果, 但 K-means 算法本身并无最优解, 实际上大部分数据挖掘算法都没有最优解, 只要分析结果有助于优化业务策略和决策支持, 分析就有意义。

## 5.4 本章小结

本章介绍了聚类分析方法论, 可以帮助读者顺畅地解决客户细分等类似问题, 但这并不意味着进行聚类分析就是小菜一碟, 聚类的实际价值很大程度上取决于变量, 而这也是方法所无能为力的。在具体分析之前, 需要对准备划分的类别有一个大致概念, 即所分的类中, 大概会有哪些特性? 又准备用怎样的变量进行类别的刻画? 有了这样的思考之后, 变量的准备就会有针对性, 一定程度上可以避免群组分不开、群组无法解释等问题。笔者认为, 聚类分析终归只是帮助我们实现分群, 并未产生新的知识, 因此不适合作为一个分析课题的核心算法或分析套路, 而应作为一种针对特定数据场景的解决方法, 或者其他分析的中间过程, 例如在客户群细分的基础上构建预测模型, 更好地去解决复杂的分析需求。

图 6-1 电商购物页面 1 (来源: 京东)

当把产品加入购物车的时候, 购物车下面会有一些推荐商品, 在这个网站的页面上有三个, 如图 6-2 所示。

推荐商品是跟手机相关的配件, 如手机的钢化玻璃膜、保护套、充电器、蓝牙耳机, 都是一些经常和手机一起使用的配套产品, 忍不住勾选了钢化膜和蓝牙耳机。

图 6-2 电商购物页面 2 (来源: 京东)

图 6-3 电商购物页面 3 (来源: 京东)

图 6-4 电商购物页面 4 (来源: 京东)

图 6-5 电商购物页面 5 (来源: 京东)

图 6-6 电商购物页面 6 (来源: 京东)

图 6-7 电商购物页面 7 (来源: 京东)

图 6-8 电商购物页面 8 (来源: 京东)

图 6-9 电商购物页面 9 (来源: 京东)

图 6-10 电商购物页面 10 (来源: 京东)

图 6-11 电商购物页面 11 (来源: 京东)

图 6-12 电商购物页面 12 (来源: 京东)

图 6-13 电商购物页面 13 (来源: 京东)

图 6-14 电商购物页面 14 (来源: 京东)

图 6-15 电商购物页面 15 (来源: 京东)

图 6-16 电商购物页面 16 (来源: 京东)

图 6-17 电商购物页面 17 (来源: 京东)

图 6-18 电商购物页面 18 (来源: 京东)

图 6-19 电商购物页面 19 (来源: 京东)

图 6-20 电商购物页面 20 (来源: 京东)

图 6-21 电商购物页面 21 (来源: 京东)

图 6-22 电商购物页面 22 (来源: 京东)

图 6-23 电商购物页面 23 (来源: 京东)

图 6-24 电商购物页面 24 (来源: 京东)

图 6-25 电商购物页面 25 (来源: 京东)

图 6-26 电商购物页面 26 (来源: 京东)

图 6-27 电商购物页面 27 (来源: 京东)

图 6-28 电商购物页面 28 (来源: 京东)

图 6-29 电商购物页面 29 (来源: 京东)

图 6-30 电商购物页面 30 (来源: 京东)

图 6-31 电商购物页面 31 (来源: 京东)

图 6-32 电商购物页面 32 (来源: 京东)

图 6-33 电商购物页面 33 (来源: 京东)

图 6-34 电商购物页面 34 (来源: 京东)

图 6-35 电商购物页面 35 (来源: 京东)

图 6-36 电商购物页面 36 (来源: 京东)

图 6-37 电商购物页面 37 (来源: 京东)

图 6-38 电商购物页面 38 (来源: 京东)

图 6-39 电商购物页面 39 (来源: 京东)

图 6-40 电商购物页面 40 (来源: 京东)

图 6-41 电商购物页面 41 (来源: 京东)

图 6-42 电商购物页面 42 (来源: 京东)

图 6-43 电商购物页面 43 (来源: 京东)

图 6-44 电商购物页面 44 (来源: 京东)

图 6-45 电商购物页面 45 (来源: 京东)

图 6-46 电商购物页面 46 (来源: 京东)

图 6-47 电商购物页面 47 (来源: 京东)

图 6-48 电商购物页面 48 (来源: 京东)

图 6-49 电商购物页面 49 (来源: 京东)

图 6-50 电商购物页面 50 (来源: 京东)

图 6-51 电商购物页面 51 (来源: 京东)

图 6-52 电商购物页面 52 (来源: 京东)

图 6-53 电商购物页面 53 (来源: 京东)

图 6-54 电商购物页面 54 (来源: 京东)

图 6-55 电商购物页面 55 (来源: 京东)

图 6-56 电商购物页面 56 (来源: 京东)

图 6-57 电商购物页面 57 (来源: 京东)

图 6-58 电商购物页面 58 (来源: 京东)

图 6-59 电商购物页面 59 (来源: 京东)

图 6-60 电商购物页面 60 (来源: 京东)

图 6-61 电商购物页面 61 (来源: 京东)

图 6-62 电商购物页面 62 (来源: 京东)

图 6-63 电商购物页面 63 (来源: 京东)

图 6-64 电商购物页面 64 (来源: 京东)

图 6-65 电商购物页面 65 (来源: 京东)

图 6-66 电商购物页面 66 (来源: 京东)

图 6-67 电商购物页面 67 (来源: 京东)

图 6-68 电商购物页面 68 (来源: 京东)

图 6-69 电商购物页面 69 (来源: 京东)

图 6-70 电商购物页面 70 (来源: 京东)

图 6-71 电商购物页面 71 (来源: 京东)

图 6-72 电商购物页面 72 (来源: 京东)

图 6-73 电商购物页面 73 (来源: 京东)

图 6-74 电商购物页面 74 (来源: 京东)

图 6-75 电商购物页面 75 (来源: 京东)

图 6-76 电商购物页面 76 (来源: 京东)

图 6-77 电商购物页面 77 (来源: 京东)

图 6-78 电商购物页面 78 (来源: 京东)

图 6-79 电商购物页面 79 (来源: 京东)

图 6-80 电商购物页面 80 (来源: 京东)

图 6-81 电商购物页面 81 (来源: 京东)

图 6-82 电商购物页面 82 (来源: 京东)

图 6-83 电商购物页面 83 (来源: 京东)

图 6-84 电商购物页面 84 (来源: 京东)

图 6-85 电商购物页面 85 (来源: 京东)

图 6-86 电商购物页面 86 (来源: 京东)

图 6-87 电商购物页面 87 (来源: 京东)

图 6-88 电商购物页面 88 (来源: 京东)

图 6-89 电商购物页面 89 (来源: 京东)

图 6-90 电商购物页面 90 (来源: 京东)

图 6-91 电商购物页面 91 (来源: 京东)

图 6-92 电商购物页面 92 (来源: 京东)

图 6-93 电商购物页面 93 (来源: 京东)

图 6-94 电商购物页面 94 (来源: 京东)

图 6-95 电商购物页面 95 (来源: 京东)

图 6-96 电商购物页面 96 (来源: 京东)

图 6-97 电商购物页面 97 (来源: 京东)

图 6-98 电商购物页面 98 (来源: 京东)

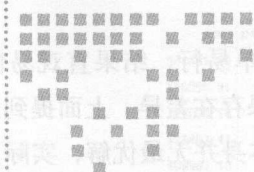
图 6-99 电商购物页面 99 (来源: 京东)

图 6-100 电商购物页面 100 (来源: 京东)

除了电商之外, 有些银行也将产品推荐给符合条件的客户, 例如招商银行“交叉销售之王”称号的富国银行。



图 5-3 K-means 算法流程图



## Chapter 6

## 第6章

## 关联规则挖掘，发现产品加载和交叉销售机会

无，名天地始；有，名万物母。常无，欲观其妙；常有，欲观其微。此两者，同出而异名，同谓之玄。玄之又玄，众妙之门。

——《道德经》

独创性常常在于发现两个或两个以上研究对象或设想之间的联系或相似之点，而原来以为这些对象或设想彼此没有关系。

——澳大利亚科学家 贝弗里奇

马克思主义认为，世界万物是联系的、发展的，并且这种联系是多样的、普遍存在的。因此，人与事物都不可能孤立存在，一件事情的发生往往会引起或伴随着另一件事情的发生。当我们发现了一件事情时，就可以由此推测另外一件事情的发生，当然，前提条件是我们已经掌握了哪些事情之间存在联系。这就是数据挖掘经典方法之一的关联规则挖掘的核心价值。在众多业务领域中，交叉销售是应用关联规则提升销售成功率和客户价值贡献的典范，通过发现产品或客户行为之间的关联规则，例如有些产品经常被一起购买、代发工资日开放式理财产品旺销、购买寿险的客户往往也会购买健康险等，构建交叉销售体系，实现销售额、服务效率和客户体验的综合提升。

## 6.1 销售的真谛:让客户买得更多

### 6.1.1 案例:电商的生意经

图 6-1 是一个很常见的电商购物页面,这是一款手机,有品牌、价格和一些相关的个性化选项。

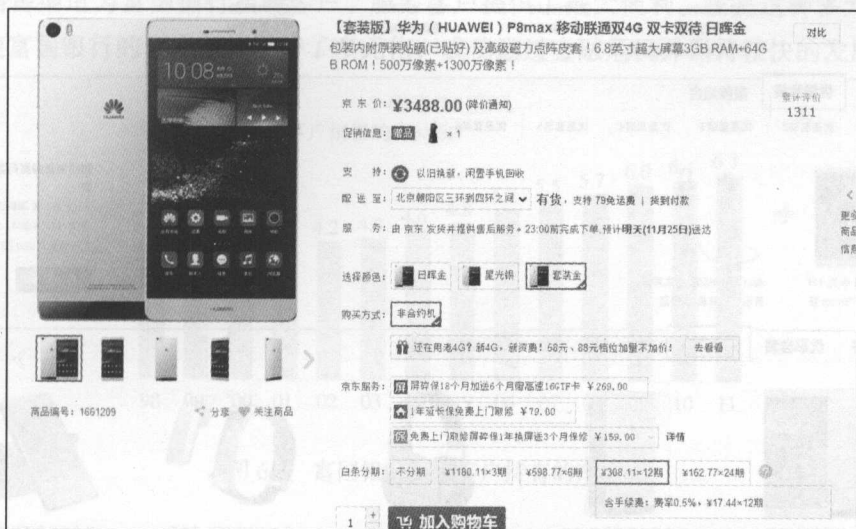


图 6-1 电商购物页面 1 (来源: 京东)

当把产品加入购物车准备购买的时候,发现购物车下面还有一些推荐栏位,在这个网站的页面上有三个,如图 6-2 所示。

□ 推荐配件: 为这个产品的关联配件,如手机的钢化玻璃膜、保护套、充电宝、蓝牙耳机,都是一些经常和手机一起使用的配套产品,忍不住勾选了钢化膜和蓝牙耳机。

□ 优惠套装: 表示再买一个运动音乐蓝牙耳机比单独买能节省 50 块钱。

□ 最佳组合: 既然别人一起使用,那么说明这两样产品的确有共通之处,值得考虑……

再往下看,有时还会发现更多栏位,例如,推荐产品(通常比正在浏览的产品更贵)、购买了这个产品的顾客也购买了、同品牌的其他产品、购买了这个产品的顾客还浏览了……

原本打算买手机的我们一次性买了手机、钢化膜、手机套、蓝牙耳机,甚至买了比原计划更贵的一款手机,先不说是否是一起购买更划算,反正是多买了很多产品。

这就是电商的交叉销售(买了更贵的产品属于向上销售,不作讨论)。

除了电商之外,有些银行也将产品销售作为重要的经营战略,其中最著名的莫过于有“交叉销售之王”称号的富国银行。



图 6-2 电商购物页面 2 (来源: 京东)

### 6.1.2 案例：富国银行的“商店”经营模式

2008 年，次贷危机让美国众多银行业巨头遭受重挫，但富国银行（Wells Fargo）却以轻微亏损的成绩单脱颖而出。2013 年 7 月，富国银行市值超越中国工商银行，成为全球市值最高的银行。

富国银行的经营战略明确而固定，即坚持做贴近普通消费者的金融零售商。为此，富国银行始终保持独特的零售商经营模式，甚至不把自己的分支机构称为支行，而是称为金融商店（store）。在金融商店中，通过系统整合实体商店、ATM、电话银行、网上银行、手机银行等渠道，为客户搭建了综合性服务网络，销售包括支票、储蓄、信用卡、保险、基金等 80 多项业务的数百种产品。与此同时，富国银行也有专卖店，只销售某项特定的业务与产品，如从事住房抵押贷款的专卖店。

富国银行的营销策略也与金融商店的称呼一致，历来重视交叉销售，并将其作为银行的重要战略，甚至把每个客户使用的产品和服务数量作为考核业绩的 KPI（关键绩效指标）。

截至2011年,富国银行单位客户销售金融产品数连续12年实现提升,同年末,公司与零售板块平均每位客户持有金融产品的数量已超过6个(见图6-3)。富国银行设定的向单位客户销售金融产品的目标是8个,而据公司估计这个数目最多可达到14个到16个。

目前,富国银行80%的业务和盈利增长都来自于向现有客户的交叉销售,因此也在业内赢得了“交叉销售之王”的称誉。同时,9000多个“商店”、12000多台庞大的自助设备、健全的销售渠道为富国银行接触客户、服务客户提供了最大便利。正是这种多元化的销售渠道,使富国银行的客户群体保持了多元化,成功逃过金融危机并保持较快的发展态势。

平均向每个零售客户销售的产品数

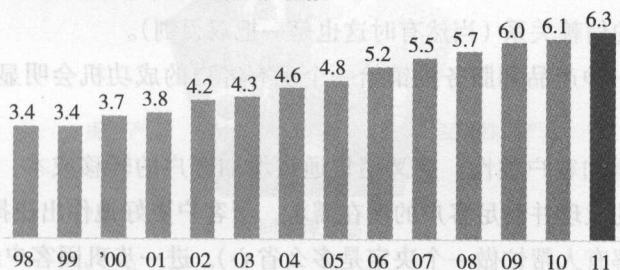


图6-3 富国银行每个客户持有的产品数

### 6.1.3 案例总结

这两个案例虽然行业不同,但有很多共通之处。

1) 在销售和服务过程中是以客户为中心,而不是以产品为中心的。销售的表现形式就是从客户的实际需求出发,通过分析大部分人的产品使用与购买习惯,为客户提供更多的选择或借鉴,由客户自己决定是否购买,而不是像某些行业那样将几种固定的产品以反复游说的方式“强推”给客户,根本不考虑客户需不需要。

2) 重视维护发现并维护优质客户资源。在现有的优质客户中发现其共同特点,并寻找符合这些特点的潜在优质客户,而后在产品的设计上以交叉销售和使用为目标不断创新,提高单个订单价值,同时让购买多项产品的客户获得价格优惠和使用上的便利。

3) 建立一套有效的交叉销售管理系统,如电商的个性化推荐系统和富国银行的立体化金融商店服务网络。把与客户的每一次接触都转化为一个销售机会,同时由于提高了单个订单价值,大大提高了对客户资源的利用能力,实现了效益的最大化。

它们都是交叉销售的典范,并且通过重视交叉销售获得了商业上的成功。



## 6.2 交叉销售

交叉销售 (cross selling) 是指借助客户关系管理, 发现客户的多种需求, 并通过满足其需求而销售多种相关服务或产品的一种营销方式, 从横向角度开发产品市场。

一句话, 是向购买 A 产品的客户推销 B 产品。

### 6.2.1 为什么要做交叉销售

其一, 销售更容易成功。交叉销售大多数时候是针对现有客户的, 而将一种产品和服务推销给一个现有客户比推销给新客户更容易。一方面是因为对现有客户有更多理解, 同时也得益于已建立的信赖关系 (当然有时这也是一把双刃剑)。

实践证明, 将一种产品和服务推销给一个现有客户的机会明显高于推销给一个新客户, 约为 5 倍。

其二, 有助于增加客户黏性。交叉销售通过增加客户的转移成本, 从而增强客户的忠诚度; 依赖更精准的发现并满足客户的潜在需求, 帮客户更好地作出抉择 (想想网购时的眼花缭乱, 就不难理解有人帮忙做一个决定是多么省心), 进一步巩固客户的依赖和信任关系。

经济学上有一个经验, 持有两项产品的客户流失率达 55%, 而持有四种产品以上的客户流失率几乎为 0。

其三, 有助于增强盈利能力。通过交叉销售可以有效地控制费用和降低成本, 提高单个客户的贡献度, 从而获得更高的盈利能力。

根据 Forrester 研究公司分析专员 Sucharita Mulpuru 的研究成果, 在电商行业, 向顾客推荐产品所产生的价值肩负起了给电商网站带来 10% ~ 30% 的收益的重任。亚马逊在 2006 年曾宣称其公司 35% 的收益都是通过向上销售和交叉销售实现的。

中国平安是国内最负盛名的交叉销售专家, 车险保费收入的半数以上来自交叉销售和电话销售渠道, 平安银行新发行的信用卡中的三分之一、新增零售存款中的四分之一均来自交叉销售渠道。

### 6.2.2 怎么做交叉销售

采取什么方式进行交叉销售, 很多时候要考虑产品之间的关系。通常将产品概括为以下五种关系, 如图 6-4 所示。

□ 竞争性的产品。同样类型和参数 (多数时候价位也相同) 但是品牌不同的产品, 如上海通用的君越和大众的帕萨特, 华为 P8 手机与 iPhone 6。

□ 互补性的产品。产品之间没有竞争性, 相反在功能上能够互相补充, 比如手机与贴膜, 上衣和裤子、衬衫、皮带等。

- 同品牌产品。同一品牌的不同产品,对品牌控们特别有帮助,最具代表性的当属苹果公司的产品线,如 iPhone、iPad、Mac Book Air、iWatch 等。
- 配件产品。为产品关联的配件与延伸服务,和互补性产品的功能补充不同,配件产品是一起使用的,比如轿车与保险、保修服务,手机和大容量存储卡。
- 价格相似的产品。价格相似、品牌不同的同个产品,适合于严格控制预算但对品牌无要求的客户,可以一起推荐给客户作对比。



图 6-4 产品之间的五种关系

根据产品之间的关系以及实现方式,有三类交叉销售最具代表性,如图 6-5 所示。

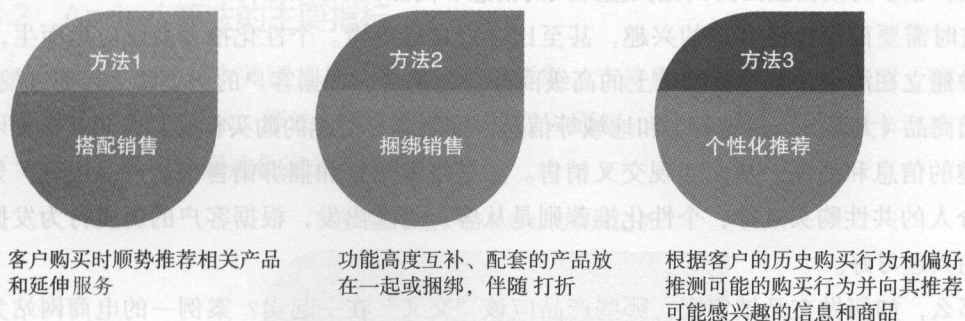


图 6-5 交叉销售的典型方式

### 1. 搭配销售

当客户购买某一个产品的时候,顺势推荐一些相关的产品或者延伸服务,往往会吸引客户一次性购买更多。比如案例一的电商网站,每一款手机的下方都有推荐的屏幕贴膜、手机套等可以更好地使用手机的附属产品,或者屏幕破碎保险、延长免费保修期等延伸服务。

实体商户也是如此。有一次逛宜家,在准备买下一个台灯的时候,发现旁边有一张很卡通的提示贴:“亲爱的顾客,灯泡是单卖的,再买一个最新款的  $\times \times$  LED 灯吧,跟这款台灯风格很搭”。

有非权威数据表明，在产品旁边适当地加入风险提示或者服务方面的提示，可以卖出更多的搭配产品。

## 2. 捆绑销售

这可能是最早被商家使用和司空见惯的交叉销售方式了。去超市购物的时候经常会发现一些商品被胶带捆在了一起，一大桶酱油带着一小瓶芝麻油、一瓶大的鲜奶盒上粘着一杯酸奶……另外，还记得案例一电商网站的优惠套装吗？

这些都属于捆绑销售，指的就是当两种或更多种产品彼此功能互补、配套的时候，在实体店或网站上将它们摆在一起，建议客户成套购买。捆绑销售的商品往往都伴随着打折，既让客户多买，又节省了他们寻找捆绑产品的时间，让人感觉贴心，真是一举多得。

## 3. 个性化推荐

有一次笔者在某个电商网站上看了几款皮靴，后来犹豫不定没有下单。第二天早上打开电脑，跳出来一个广告窗，全是我喜欢的皮靴款式，还有几款皮带，忍不住就点了进去（别忘了网站的三大关键指标：入口流量、用户转化率、平均订单价值）。

这是一个商品极丰富、信息泛滥的时代，实体店铺里的货架琳琅满目，电商网店上面的栏位层峦叠嶂，最耗精力和脑细胞的不是纠结于要不要买，而是怎样找到自己想要购买的商品，很多时候甚至因为浏览大量无关的信息和商品而放弃购买。

这时需要商家读懂客户的兴趣，甚至比客户自己更懂。个性化推荐系统应运而生，这是一种建立在海量数据挖掘基础上的高级商务智能平台，根据客户的历史购买行为、偏好、最热卖商品（大部分人的选择）和地域等信息，推测客户可能的购买行为，并向其推荐可能感兴趣的信息和商品，从而实现交叉销售。如果搭配销售和捆绑销售是从产品出发，分析大部分人的共性购买规律，个性化推荐则是从客户需求出发，根据客户的历史行为发掘其独特的兴趣习好。

那么，如何做交叉销售呢，哪些产品应该“交叉”在一起卖？案例一的电商网站为什么把手机和蓝牙耳机作为优惠套装，而不是手机和数据线或者智能手环，超市为何把酱油和芝麻油捆绑在一起，而不是酱油和牛奶？答案是看哪些产品之间关联规则强。

## 6.3 关联规则挖掘，发现交叉销售机会

要发现哪些产品经常被一起购买，从而可以交叉销售或推荐，方法有很多种。其中一种最直接的方法是针对购买“某种产品”的全部客户，统计他们同时购买最多的产品是哪些，当然可以统计得更细，例如同时购买最多的互补产品、配件等。更多时候，我们并不针对具体的“某种产品”，而是从海量的购买记录中直接寻找交叉销售的机会。



作为最具影响力的挖掘关联规则的数据挖掘算法, 网上有人归纳了十大机器学习算法, Apriori 算法位列其中。作为最具影响力的挖掘关联规则的数据挖掘算法, Apriori 算法已经被广泛地应用到购物篮分析 (交叉销售的一种)、商场顾客的消费轨迹、网络入侵检测等多个领域。

### 6.3.1 Apriori 算法

Apriori 算法是一种挖掘布尔关联规则频繁项集的算法, 使用候选项集通过设定一些指标找出频繁项集。项集就是产品的任意组合, 频繁项集就是经常被一起购买的产品组合, 反映了大部分人的购买习惯。简言之, Apriori 算法就是要发现大部分客户一起购买的产品 X、Y, 建立  $X \rightarrow Y$  或者  $Y \rightarrow X$  的关系式。

Boolean (布尔运算) 通过对两个以上的物体进行并集、差集、交集的运算, 从而得到新的物体形态, 感兴趣的读者可查阅相关资料。

算法的基本思想如下。

1) 如果一个项集不是频繁项集, 那么任何包含它的项集也一定不是频繁项集。也就是说, 同时购买产品 X 和 Y 的人很少, 那么同时购买产品 X、Y 和 Z 的人就更少了。

2) 如果一个项集是频繁项集, 那么它的任何子集也是频繁项集。也就是说, 购买 X、Y、Z 的人多, 自然购买 X 和 Y 的人就更多了。

### 6.3.2 Apriori 算法的主要指标

Apriori 算法有五个关键指标: 项集 / 频繁项集、支持度、置信度、提升度、强关联规则, 掌握了这几个指标, 算法的运行步骤也就迎刃而解。图 6-6 是一份信用卡客户的产品持有记录, 下面将根据这份记录介绍指标。

一份信用卡客户持有产品的明细记录

钱一: 卡无忧、现金宝、任意贷、实时通  
丁二: 现金通、卡无忧、外汇通  
张三: 理财通、卡无忧、现金宝、余额盈  
李四: 余额盈、随意分  
王五: 分期宝、理财通  
赵六: 随意分、卡无忧、现金宝  
xx: ... ..

图 6-6 客户的产品持有记录 (样例)

\* 产品名称与持有情况根据案例需要设计, 并非任何公司的真实产品



### 指标 1 项集 / 频繁项集

项集是所有产品及其任意组合，一个项即为一个产品或者组合，在多数客户的订单中频繁出现的那些项就是频繁项集。

以其中的四个产品为例罗列它们的全部组合。图 6-7 中每一个元素是一个项，多个元素组成项集。其中，“卡无忧”在多个客户的订单中出现，为一个频繁项，“卡无忧、现金宝”也可能是一个频繁项，所有这些频繁项组成频繁项集。

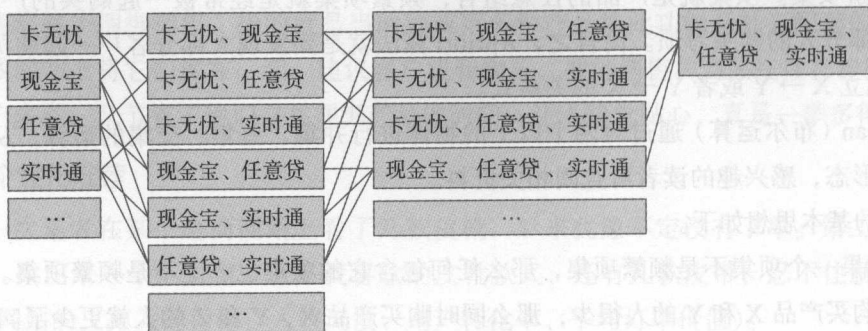


图 6-7 四个产品的全部组合

### 指标 2 支持度

怎样才算频繁？依据项在全部客户记录中出现的频率，或称为支持度。支持度体现了关联规则的普遍性。

支持度的计算方法：包含某个项的订单数 / 全部订单数。例如，“卡无忧”的支持度 = 4/6，即 6 个人当中有 4 个人持有；“卡无忧 → 现金宝”的支持度 = 3/6，即 6 个人当中有 3 个人同时持有这两个产品。

### 指标 3 置信度

对于那些经常被一起购买的产品，它们之间存在较强的关联性。怎么定义关系的强弱？这时会用到置信度指标。

置信度是指客户在持有一个产品（或者一组产品）的前提下，还持有其他产品的概率。置信度体现关联规则的可靠性。

例如，“卡无忧 → 现金宝”的置信度 = 3/4，即持有“卡无忧”的 4 个客户当中有 3 个人同时持有“现金宝”。

### 指标 4 提升度

持有左侧产品的客户持有右侧产品的置信度，相比右侧产品支持度的倍数，体现规则的实际价值。也可以理解为：满足条件的事件相比不满足条件的事件，发生增加的可能性。

计算公式： $\text{Lift}(A \rightarrow B) = \text{CONF}(A \rightarrow B) / \text{SUPP}(B)$

例如,  $\text{Lift}(\text{“卡无忧} \rightarrow \text{现金宝”}) = 1.5$ , 即  $3/4$  除以  $3/6$ 。

### 指标 5 强关联规则

诸如  $A \rightarrow B$ , 购买 A 的情况下又购买 B 的事件, 称为关联规则。其中, 满足最小支持度阈值和最小置信度阈值的关联规则称为强关联规则 (规则既普遍, 又可靠)。

例如, 如果设定最小支持度阈值 = 30%, 最小置信度阈值 = 70%, 那么 “卡无忧  $\rightarrow$  现金宝” 就是一条强关联规则, 因为它的支持度和置信度分别为 50%、75%, 大于阈值。

### 6.3.3 Apriori 算法的基本步骤

Apriori 算法主要包括如下几个步骤 (见图 6-8)。

1) 数据输入: 按照格式要求准备数据, 一般包含客户 ID 和产品名称两列数据。可同时设定支持度阈值、置信度阈值。

2) 生成项集: 根据出现的全部产品, 生成全体项集。

3) 计算项集的支持度: 针对每个项, 分别计算支持度, 同时去除小于支持度阈值的项, 得到频繁项集。

4) 计算关联规则的置信度: 针对频繁项集, 建立诸如  $A \rightarrow B$  的全部关联规则, 计算置信度, 同时去除小于最小置信度阈值的规则。

5) 将强关联规则直观展示出来。

6) 规则分析: 从业务角度出发分析规则的实际意义, 提取具有现实指导意义的关联规则。

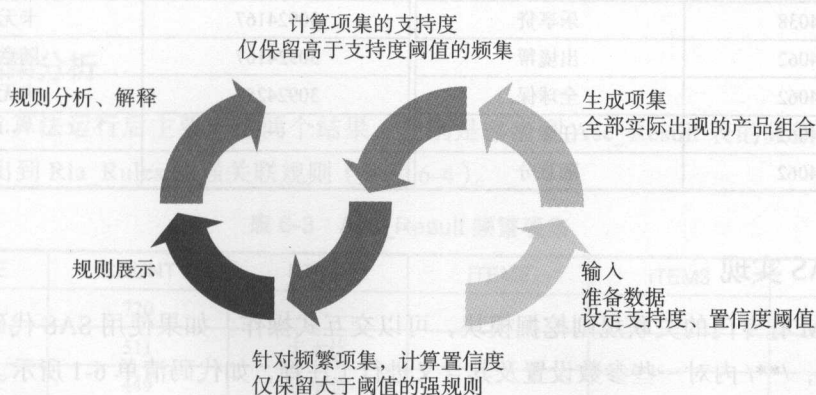


图 6-8 Apriori 算法的基本步骤

## 6.4 案例: 信用卡产品交叉销售

本节结合某银行信用卡公司的产品交叉销售案例, 介绍如何应用 SAS 实现 Apriori 算

法，挖掘产品之间的关联规则，侧重实际的分析过程和需要注意的地方，不拘泥于具体的技术细节。

### 6.4.1    准备数据

每个数据挖掘算法对于要分析的数据都有特定的格式要求，Apriori 算法要求数据包含客户 ID 和产品名称两列。表 6-1 是一份信用卡客户持有产品的清单，类似的还有超市购物单、网店订单等。要应用 Apriori 算法挖掘关联规则，则需要转换成表 6-2 所示的格式。

表 6-1    信用卡客户持有产品的清单（样例）

Csr_Id	Prod
30924038	全球保、随意分、信用保、乐享贷
30924062	出境帮、全球保、任意贷、随意分、信用保
30924167	实时通、出境帮、卡无忧、随意分
30924255	卡无忧
...	...

注：产品名称与持有情况应根据案例需要设计，并非任何公司的真实产品。

表 6-2    Apriori 算法的输入数据格式

Csr_Id	Prod	Csr_Id	Prod
30924038	全球保	30924062	信用保
30924038	随意分	30924167	实时通
30924038	信用保	30924167	出境帮
30924038	乐享贷	30924167	卡无忧
30924062	出境帮	30924167	随意分
30924062	全球保	30924255	卡无忧
30924062	任意贷	...	...
30924062	随意分		

### 6.4.2    SAS 实现

SAS EM 有专门的关联规则挖掘模块，可以交互式操作。如果使用 SAS 代码，一般包括三段代码，/\*\*/ 内对一些参数设置及其含义进行了注释，如代码清单 6-1 所示。

**Step1** 为建模创建事务数据库，可理解为分析准备一个虚拟环境，并不产生分析结果。在安装 SAS 的时候应该勾选 SAS EM 模块，否则会报错。

**Step2** 用于生成项集并根据支持度阈值得到频繁项集，当产品项很多时，运算量会非常大。此处，频繁项集输出到表 Asc\_Result 中。

**Step3** 可利用频繁项集构造满足最小置信度阈值的强关联规则，并直观展示出来，此处

输出到表 Rla\_Rules 中。

代码清单 6-1

```

/*Step1 为建模创建事务数据库 */
PROC DMBDB
DATA = Asso_Datsrc /*Asso_Datsrc 是表 2 的输入数据 */
OUT = _null_
DMBCAT=Tmp2;
ID Csr_Id; /* 指定 Csr_Id 为客户编号 */
CLASS Prod; /* Prod 为字符类型的变量 */
TARGET Prod; /* 指定 Prod 为分析关联规则的产品项 */
RUN;
/*Step2 用于生成频繁项集 */
PROC ASSOC
DMBCAT = TMP2
DATA = Assoc_Datsrc
OUT = Asc_Result /* 将频繁项集输出到 Asc_Result 表中 */
PCTSUP = 10 /* 设定支持度阈值为 10% */
ITEMS = 4; /* 最大项数为 4，即在一条规则中最多包含 4 个产品 */
CUSTOMER Csr_Id;
TARGET Prod;
QUIT;
/*Step3 生成强关联规则并直观展示出来 */
PROC RULEGEN
IN = Asc_Result /* 以上一步生成的频繁项集作为输入 */
OUT = Rla_Rules /* 将关联规则输出到表 Rla_Rules 中 */
MINCONF = 60; /* 设定置信度阈值为 60% */
QUIT;

```

### 6.4.3 结果分析

Apriori 算法运行后主要产生两个结果，分别是输出到 Asc\_Result 表的频繁项集（见表 6-3）和输出到 Rla\_Rules 的强关联规则（见表 6-4）。

表 6-3 Asc\_Result 频繁项集

SET_SIZE	COUNT	ITEM1	ITEM2	ITEM3	ITEM4
0	720				
1	511	卡无忧			
1	449	乐享贷			
1	176	随时付			
1	102	分期宝			
1	94	全球保			
1	88	财务通			
1	83	实时通			



(续)

SET_SIZE	COUNT	ITEM1	ITEM2	ITEM3	ITEM4
1	65	任意贷			
1	56	加速积分			
1	55	现金宝			
2	304	乐享贷	卡无忧		
2	144	随时付	卡无忧		
2	105	乐享贷	随时付		
2	78	分期宝	卡无忧		
2	75	实时通	全球保		
2	72	全球保	卡无忧		
2	71	财务通	卡无忧		
2	63	实时通	卡无忧		
2	52	任意贷	卡无忧		
2	52	乐享贷	全球保		
3	81	乐享贷	随时付	卡无忧	
3	59	实时通	全球保	卡无忧	

表 6-3 中各参数介绍如下。

SET\_SIZE 为项数，即一个项中产品的数量，0 表示全部人数，并非不持有产品的人数。

COUNT 表示持有对应产品的人数，这里一共有 720 个客户，其中 511 人持有“卡无忧”、304 人同时持有“乐享贷”和“卡无忧”，依此类推。

ITEM1~ITEM4 为产品项，这里 ITEM4 为空表示包含 4 个产品的项集支持度都小于 10%，即持有 4 项产品的客户占比低于 10%。

需要说明的是，此处以 511 作为计算支持度的分母，而不是 720，以使最大支持度为 100%。

表 6-4 中各参数介绍如下。

RULE 为关联规则，持有（购买）左侧产品的客户同时持有右侧产品。

COUNT 为持有规则中全部产品项的人数。

SUPPORT 为规则的支持度，持有规则中全部产品项的客户占比。例如  $\text{SUPPORT}(\text{乐享贷} \Rightarrow \text{卡无忧}) = \text{COUNT}(\text{乐享贷} \& \text{卡无忧}) / 720 = 304 / 720 = 42\%$ 。说明一下，这里计算支持度的分母为全部人数 720。

CONF 为规则的置信度，例如  $\text{CONF}(\text{乐享贷} \Rightarrow \text{卡无忧}) = \text{COUNT}(\text{乐享贷} \& \text{卡无忧}) / \text{COUNT}(\text{乐享贷}) = 304 / 449 = 68\%$ 。

EXP\_CONF 为期望置信度，即“ $\Rightarrow$ ”右侧产品（或称导出项）的支持度。例如  $\text{EXP\_CONF}(\text{乐享贷} \Rightarrow \text{卡无忧}) = \text{SUPPORT}(\text{卡无忧}) = 71\%$ 。意思是如果不通过条件限定，那

表 6-4 Rla\_Rules 强关联规则

SET_SIZE	EXP_CONF	CONF	SUPPORT	UFT	COUNT	RULE	_LHAND	_RHAND	ITEM1	ITEM2	ITEM3	ITEM4
1	71		71		511	卡无忧	卡无忧		卡无忧			
1	62		62		449	乐享贷	乐享贷		乐享贷			
1	24		24		176	随时付	随时付		随时付			
1	14		14		102	分期宝	分期宝		分期宝			
1	13		13		94	全球保	全球保		全球保			
1	12		12		88	财务通	财务通		财务通			
1	12		12		83	实时通	实时通		实时通			
1	9		9		65	任意贷	任意贷		任意贷			
1	8		8		56	加速积分	加速积分		加速积分			
1	8		8		55	现金宝	现金宝		现金宝			
2	71	68	42	1.0	304	乐享贷 ==> 卡无忧	乐享贷	卡无忧	乐享贷		卡无忧	
2	71	82	20	1.2	144	随时付 ==> 卡无忧	随时付	卡无忧	随时付		卡无忧	
2	71	76	11	1.1	78	分期宝 ==> 卡无忧	分期宝	卡无忧	分期宝		卡无忧	
2	13	90	10	6.9	75	实时通 ==> 全球保	实时通	全球保	实时通		全球保	
2	12	80	10	6.9	75	全球保 ==> 实时通	全球保	实时通	全球保		实时通	
2	71	77	10	1.1	72	全球保 ==> 卡无忧	全球保	卡无忧	全球保		卡无忧	
2	71	81	10	1.1	71	财务通 ==> 卡无忧	财务通	卡无忧	财务通		卡无忧	
2	71	76	9	1.1	63	实时通 ==> 卡无忧	实时通	卡无忧	实时通		卡无忧	
2	71	80	7	1.1	52	任意贷 ==> 卡无忧	任意贷	卡无忧	任意贷		卡无忧	
3	71	77	11	1.1	81	乐享贷 & 随时付 ==> 卡无忧	乐享贷 & 随时付	卡无忧	乐享贷	随时付		卡无忧
3	10	71	8	7.1	59	实时通 ==> 全球保 & 卡无忧	实时通	全球保 & 卡无忧	实时通		全球保	卡无忧
3	9	63	8	7.2	59	全球保 ==> 实时通 & 卡无忧	全球保	实时通 & 卡无忧	全球保		实时通	卡无忧
3	71	79	8	1.1	59	实时通 & 全球保 ==> 卡无忧	实时通 & 全球保	卡无忧	实时通	全球保		卡无忧
3	13	94	8	7.2	59	实时通 & 卡无忧 ==> 全球保	实时通 & 卡无忧	全球保	实时通	卡无忧		全球保
3	12	82	8	7.1	59	全球保 & 卡无忧 ==> 实时通	全球保 & 卡无忧	实时通	全球保	卡无忧		实时通

注：这里的强关联规则仅为帮助理解分析方法，并非真实的规律。

么该产品会有多少人购买。

LIFT 就是规则的强度了，等于规则的置信度比上期望置信度。例如  $LIFT(\text{乐享贷} \Rightarrow \text{卡无忧}) = \text{CONF}(\text{乐享贷} \Rightarrow \text{卡无忧}) / \text{EXP\_CONF}(\text{乐享贷} \Rightarrow \text{卡无忧})$ 。LIFT 小于 1，说明符合条件的客户反而不会购买右侧产品。

### 6.4.4 序列关联分析

如果非要在产品前面加上一列时间，当签约产品存在必然的先后顺序时，这时会用到序列关联分析。表 6-5 是信用卡客户的产品持有记录及其签约时间。

表 6-5 信用卡客户签约产品的明细记录

Csr_Id	Prod	Date
30924038	任意贷	26May2014
30924038	实时通	16Nov2013
30924038	随意分	21Mar2014
30924038	现金宝	14Apr2014
30924062	分期宝	26Feb2014
30924062	实时通	11Nov2013
30924062	全球保	06Sep2013
30924062	随意分	28Mar2014
30924062	现金宝	25Apr2014
...	...	

代码清单 6-2 给出了序列关联分析的 SAS 代码。在分析之前，需要先针对每个客户购买产品的先后顺序生成一系列序号，即前面两段代码的目的。找找看，与 Apriori 算法的代码有何区别？

代码清单 6-2

```
PROC SORT DATA=Sequence_Datsrc;
  BY Csr_Id Date;
RUN;
DATA Sequence_Datsrc;
  SET Sequence_Datsrc;
  BY Csr_Id Date;
  IF FIRST.Csr_Id THEN X=0;
  X+1;
RUN;

PROC DMDB DATA =sequence_datsrc
  OUT = TMP1
  DMDBCAT=TMP2;
  ID CSR_ID X;
```

```

CLASS PROD;
TARGET PROD;
RUN;

PROC ASSOC DMDBCAT=TMP2
  DATA= SEQUENCE_DATSRC
  OUT=ASC_Result
  PCTSUP = 10 ITEMS=4;
  CUSTOMER CSR_ID ;
  TARGET PROD ;
QUIT;

proc sequence dmdbcat=TMP2
  data=TMP1
  assoc=ASC_Result
  out=RLA_RULES
  ITEMS=3;
  cust CSR_ID;
  target PROD;
  visit X;
quit;

```

关于序列关联分析的参数设置和分析结果不再详述，感兴趣的读者可以运行 SAS 代码，结合输出结果进行深入了解。

### 6.4.5 结果应用

输出到表中的频繁项集和强关联规则只是算法运行的结果，很多时候并不会直接应用，需要更好地解读和展示。一般来说，关联规则挖掘的结果主要有三个。

其一，产品渗透率，直观展示产品销售现状（见图 6-9）。



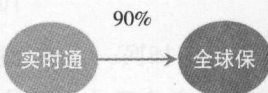
图 6-9 关联规则挖掘的第 1 个结果：产品渗透率



其二，经过业务解读并且建议投入应用的强关联规则（见图 6-10）。

交叉销售指引（样例）

- 1 向持有“实时通”的客户推荐“全球保”，预计成功率为90%  
（12%的客户持有“实时通”，其中有90%同时持有“全球保”）



- 2 向同时持有“全球保”和“卡无忧”的客户推荐“实时通”，预计成功率为82%



MORE...

图 6-10 关联规则挖掘的第 2 个结果：强关联规则及解读

其三，交叉销售客户清单。如果规则被采用，网上可以建立推荐栏位，线下则可以开展产品的定向营销，很多时候取决于当前销售的目标产品是什么（这就是规则的实际价值，如果右侧产品不是销售目标，那么规则的置信度和 LIFT 再高也没有意义）。例如，当前需要加大“全球保”的销售力度，就可以应用规则 1 筛选出持有“实时通”但尚未签约全球保的客户清单，同时提供手机号、客户名称、性别等信息，以便安排营销资源进行定向营销。

当然，关联规则也可以应用于改进网银页面，根据客户的产品持有和操作情况投放个性化的产品广告。

## 6.5 本章小结

Apriori 算法的支持度和置信度阈值很难合理设定。如果把支持度和置信度设置得足够低，将会生成非常多的规则，甚至两条规则互相矛盾；如果设置得过高，生成的规则可能又太少，有可能遗漏一些关键的规则。可以说，这两个参数的设置并没有最优解。

规则的实际价值需要用户主观确定。生成的规则并不能直接应用，其中存在许多不准确、无实际意义的规则，需要用户结合业务知识和需要对规则进行解读，从中挑选那些可以投入使用的规则。从这点上说，支持度和置信度阈值的设置有一个从低到高不断尝试的过程。

分析者可以指定对哪些数据进行挖掘。当数据量过大，特别是产品项非常多时，运算过程将会很长，这种情况下建议先行删除那些稀疏的产品和购买记录。

## 第7章

## Chapter 7

社交网络分析，从“关系”  
的角度分析问题

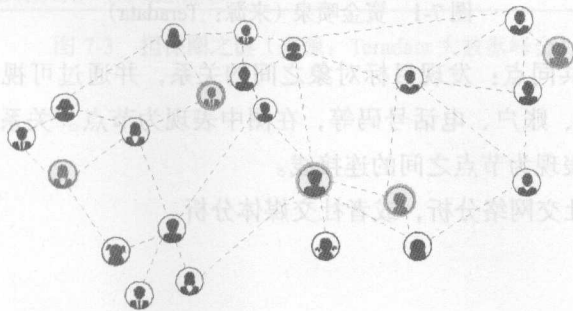
千丈之堤，以蝼蚁之穴溃；百尺之室，以突隙之烟焚。

——韩非《韩非子·喻老》

“实事”就是客观存在着的一切事物，“是”就是客观事物的内部联系，即规律性，“求”就是我们去研究。

——毛泽东

在好莱坞流行这样一句话：“一个人能否成功，不在于你知道什么（what you know），而在于你认识谁（who you know）”。埃米尔·涂尔干（Emile Durkheim, 1879）说：“有一种观点认为，对于社会生活的解释，不应当靠参与者的观念进行，而应当根据尚未被自觉认识到的更深层的原因进行。笔者认为，这样做会极富成就感。笔者还认为，其中的原因主要应当在由个人形成的组群方式中寻找。”



以上这些都是社交网络分析 (Social Network Analysis, SNA) 的思想。社交网络分析是研究一组行动者关系的方法, 一组行动者可以是人、社区、群体、组织、国家等, 其关系模式反映出现象或数据是网络分析的焦点。因此, 社交网络分析关注的焦点是关系和关系的模式, 采用的方式和方法从概念上有别于传统的统计分析和数据处理方法。十年前, IBM 公司使用这种方法研究团队建设和知识管理, 在当时绝无仅有。而现在, 这种分析越来越时髦, 其研究思路在众多业务领域中得以应用, 并成为大数据分析的重要方向。

## 7.1 先看几张美轮美奂的图片

一次参加 Teradata 主办的大数据峰会, 在会场外展示了一组画, 这组画是多种可视化分析的图案, 很有艺术气息, 而且透过图案对业务进行了深邃的洞察。下面为读者介绍其中的几幅 (见图 7-1 ~ 图 7-4)。

### 资金喷泉

#### 业务洞察

分析大型企业之间的资金流动量关系, 银行根据企业的转账交易数据, 了解风险, 发现市场机会。在此图中, 节点代表公司, 线代表两家公司间的资金转移, 箭头显示这笔钱的流向。图中显示了不同公司之间的所有资金流动, 可以了解上下游供应链关系和对彼此相互依赖的公司。

管理风险。银行借此识别出高度依存的公司, 确定关键的公司, 在供应链和独立交叉检查公司的现金流来验证其资金的健康状况。识别欺诈。银行可以检查公司真正的商业活动, 并可以验证借出资金的用途。例如制造商, 是投资在股票市场的投机资金, 而不是向供应商付款或谁拿出一笔贷款建一座工厂, 但真正的资金将用于短期住宅房地产交易。

市场营销。大量资金流出的链标识潜在的高价值公司, 为现有客户端揭示了在融资、结算、风险管理等更广泛的金融服务提供更高的价值。

#### 数据

通过对超过 67 万个公司的 6080 万条交易记录进行分析, 构建资金往来图, 该图选择了其中 32 个最重要的公司及其主要交易对手 (交易量超过 70 万元)。

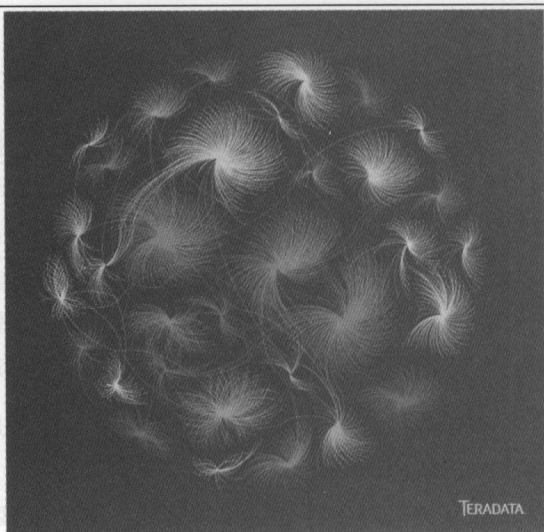


图 7-1 资金喷泉 (来源: Teradata)

这些分析有一个共同点: 发现目标对象之间的关系, 并通过可视化方式展示出来。目标对象可以是客户 ID、账户、电话号码等, 在图中表现为节点。关系可以是资金往来、担保、通话等, 在图中表现为节点之间的连接线。

这类分析统称为社交网络分析, 或者社交媒体分析。

### 担保圈火花

#### 业务洞察

展现了在某家银行的汽车车商、4S 店和个人客户之间建立的担保关系网络, 这是担保圈分析中的一个图形。

制造商和零售商早就给客户消费贷款来促进销售的好处。汽车行业的竞争越来越激烈, 为了更好地争取客户, 很多汽车金融公司和 4S 店为客户申请汽车消费贷款。

在这个隐去名字的图中, 点代表车贷的客户或者车贷担保人, 线代表担保人和被担保人的担保关系, 不同的颜色用于区分相应的担保网络。

从图中的这些火花绽放, 可以很容易看到黄色、蓝色和紫色的群体, 这其中几个担保人为很多贷款提供担保。这些担保人有些是汽车公司或企业法人。有些情况下, 像蓝色和紫色群体, 不同的相关实体作为担保人, 这样就很难发现总体的风险暴露。银行可以针对这些风险采取措施。首先要做的是防止多米诺效应。很多情况下, 这些汽车厂商和 4S 店是银行的优质客户, 银行出于维护客户关系, 扩大业务范围, 提高客户黏度, 为此办理车贷业务, 但相应的车贷风险还是要做好监控和管理。

#### 数据

数据源包含担保人企业 ID、担保合同信息、担保金额、企业信用评级等, 通过分析有影响力的客户来识别担保模式。

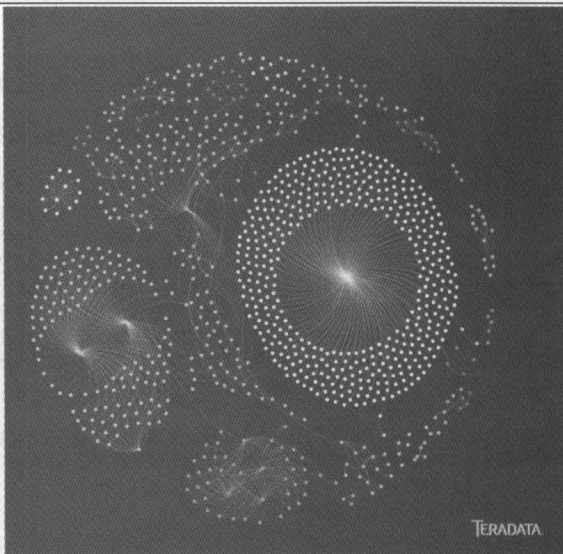


图 7-2 担保圈火花 (来源: Teradata)

### 担保圈之谜

#### 业务洞察

揭示房地产开发行业所产生的潜在风险。首先, 房地产开发商是银行的重要客户, 开发住宅需要银行的资金支持, 房屋建成后, 银行给购房者提供住房抵押贷款, 延伸和拓展了银行业务, 同时帮助房地产开发商销售产品。

在经济转型阶段, 身在淘汰落后产能的购房者出现断供的可能性增加, 如果同时房价出现下滑, 房贷风险会进一步增加。房地产开发商作为担保人, 他们未来的风险需要防范。

从银行的角度讲, 房贷是按照每个购房者的资质来审批的。每个人都房贷, 如果由房地产开发商担保, 本身的风险并不大, 但如果房地产开发商为很多人提供担保, 很可能是房子没有卖出去, 制造一种虚假销售, 套取银行贷款, 那这样的风险就要更加防范。

这个隐去名字的图形让银行看清房地产开发商和房贷客户的担保关系。每个点代表房地产开发商或房贷客户, 线是房地产开发商和房贷客户间的担保贷款, 不同颜色可以区分不同的担保网络, 你会看到一些房地产开发商为大量的客户提高贷款的担保。

“担保圈之谜”为银行真实展现和监控风险暴露的规模, 银行可以对高风险客户采取措施, 包括收紧贷款审批, 甚至拒绝贷款。

#### 数据

数据源包含担保人企业 ID、担保合同信息、担保金额、企业信用评级等, 通过分析发现有影响力的客户, 识别担保模式。

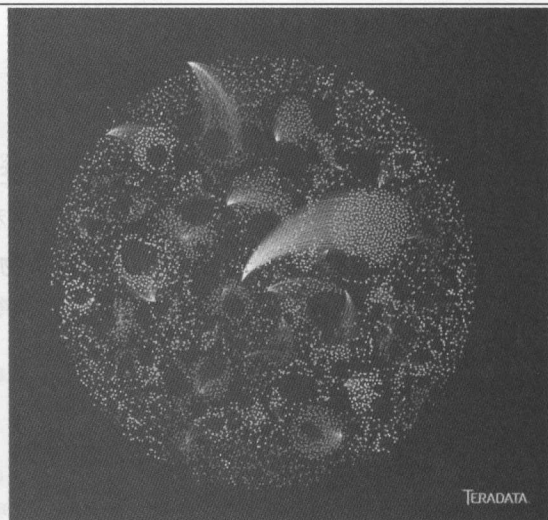


图 7-3 担保圈之谜 (来源: Teradata 大数据峰会)



### 通话圈

#### 业务洞察

无论何时何地，我们都在使用手机并产生出大量的行为和活动信息。我们与其他人的每通电话和短信都显示出我们的社会关系、商业活动以及更广泛的社群互动，并形成许多复杂的互相连接的通话圈。

图中，每个点代表一个手机号码，越大的点表示这个号码被拨打的次数越多，线表示从一个号码拨打到另一个号码。

每个号码（用户）会有一种独特的通话模式，参照这种模式可以设计合适的话费方案或者预测用户的行为。举例来说，当一个用户正要从现在的电信服务商转换到另一个服务商时，可以从网内及网外发现两种类似的通话模式。

这张图展示的是几秒钟时间内的通话模式，从左上角可以看到许多大圆圈，表示短时间内这些号码被拨打了许多次。可以推测这些号码有可能是机器，像是自动答录机、互动式语音应答 IVR 系统、安全系统或警报。人类不可能在短时间拨出这么多电话。这些电话会先放在一个分开的群组，后续的分析就可以集中在个人用户的通话模式上。

#### 数据

数据源包括拨出号码、呼入号码、拨出时间、通话时间，时间稍长这个网络就会变得异常庞大，导致不可能被视觉化，因此需要限定时间到一个可以呈现的范围。

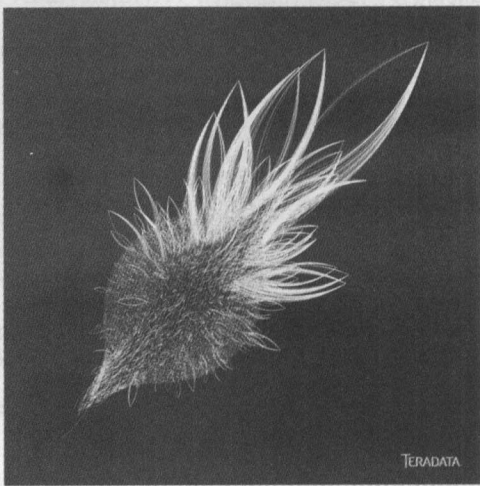


图 7-4 通话圈（来源：Teradata 大数据峰会）

## 7.2 社交网络分析方法

### 7.2.1 定义

在本章的开头提到过，社交网络分析关注的焦点是参与者的关系和关系的模式，采用的方式和方法从概念上有别于传统的统计分析和数据处理方法。

简言之，社交网络分析是发现人或者事物之间是否存在关联关系以及存在何种关系的分析方法。表示关系（例如交易、通话、担保等）的数据是分析依据，可视化图形是发现各种关系的直观方法，除此之外，图论、数学、社会学等领域为社交网络分析提供了一些理论依据和算法。

### 7.2.2 应用场景

社交网络分析的思路已成为大数据分析的重要方向，在众多业务领域中得以应用。包括 Teradata 展示的这些图形，比较典型的应用场景包括以下几方面。

- 1) 描绘信息传播路径：发现舆论领袖，创新扩散过程。
- 2) 通话圈、朋友圈：发现高影响力的大 V 用户，选择信息投放目标。
- 3) 产业链与价值链：理清产业上下游，发现潜在客户及其需求。
- 4) 风险蔓延：分析风险蔓延的路径并量化整体客户群的风险。
- 5) 知识管理与知识的传递：弱关系的力量。

### 7.2.3 网络识别算法

社交网络分析的数据源是交易流水, 无论是银行、第三方支付公司还是电商, 数据量通常都为亿万级别, 非常巨大, 涉及的企业或个人 (在图上显示为节点) 也数以万计, 通过可视化工具直接展示出来几乎不可能。这时需要指定一些初始节点, 作为生成关系网络或者供应链的出发点, 通过交易关系发现其上下游节点, 然后再以这些节点出发, 一层一层地像波浪那样拓展开去……最终形成该节点企业或个人的关系网络, 然后通过可视化软件展示出来。

需要设计一个算法, 识别出同属于一个网络的所有节点及其关系。

算法的设计思路: 把每个关系网络 (或者链条) 看成一棵树。每棵树从独立的树根开始生长, 从根开始每个节点发出多根树枝; 树枝只能正向生长, 树的生长过程会受控制, 脆弱的、旁生的树枝会被剪除, 以保证树能长大; 每根树枝都是独一无二的。

算法步骤如下。

- 1) 将每个指定的企业作为根节点。
- 2) 关联直接交易对手 (分枝, 包括收款和付款), 删除交易额小于阈值的交易记录, 每个节点客户归属于交易额最高的根节点客户。
- 3) 围绕第一层分枝节点客户, 关联交易对手, 向外扩展, 同步骤 2)。
- 4) 当达到指定的扩展层次时, 停止。

最终形成以初始节点为核心, 涵盖上下游客户的若干组关系链, 通过资金流向、物流方向等定义上下游, 图 7-5 所示的是算法原型。

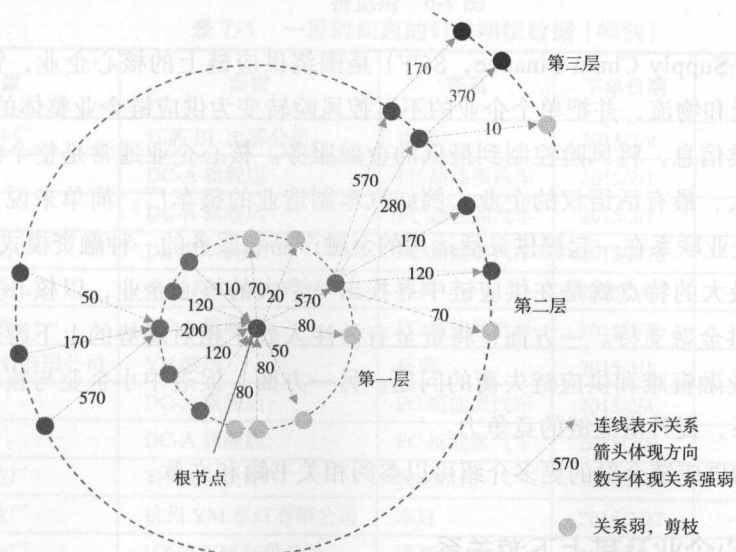


图 7-5 算法原型

下面以两个非常典型的电商业务场景作为案例，介绍社交网络分析在其中发挥的支撑作用。其一是营销领域的供应链，其二是风险管理领域的风险蔓延。

## 7.3 案例：电商通过订单数据识别供应链

### 7.3.1 供应链及供应链金融

通过对信息流、物流、资金流的控制，从采购原材料开始，到制成中间产品及最终产品，最后由销售网络把产品送到消费者手中的过程，就是供应链，将供应商、制造商、分销商、零售商、直到最终用户连成一个整体的功能网链结构，如图 7-6 所示。

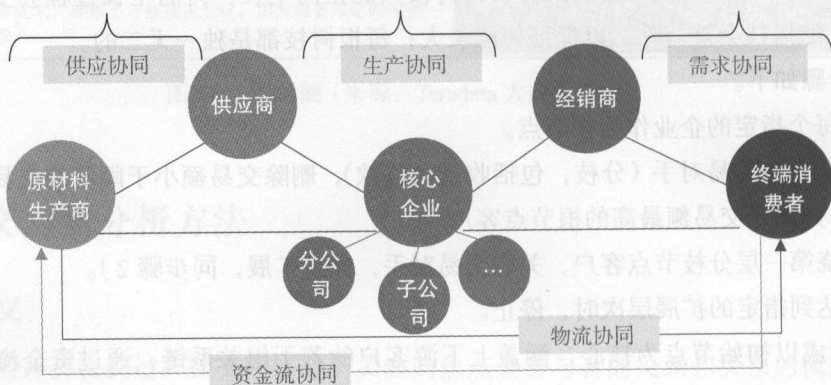


图 7-6 供应链

供应链金融（Supply Chain Finance, SCF）是围绕供应链上的核心企业，管理上下游中小企业的资金流和物流，并把单个企业的不可控风险转变为供应链企业整体的可控风险，通过立体获取各类信息，将风险控制到最低的金融服务。核心企业通常是整个链条中竞争力最强、规模最大、最有话语权的企业，例如汽车制造业的整车厂。简单来说，就是将核心企业和上下游企业联系在一起提供灵活运用的金融产品和服务的一种融资模式。

供应链金融最大的特点就是在供应链中寻找出一个大的核心企业，以核心企业为出发点，为供应链提供金融支持。一方面，将资金有效注入处于相对弱势的上下游配套中小企业，解决中小企业融资难和供应链失衡的问题；另一方面，促进中小企业与核心企业建立长期战略协同关系，提升供应链的竞争力。

关于供应链和供应链金融的更多介绍可以参阅相关书籍和文章。

### 7.3.2 识别核心企业及其上下游关系

对于电商平台来说，如何识别同属一个供应链的企业？如何判断核心企业？每个企业



的上下游在哪里?

根据订单数据, 按照社交网络分析的理念和算法, 应用企业之间的订单数据, 就可以绘制出企业间的物流和资金关系(供应链), 不仅能串起彼此存在产业供需关系的企业, 还能看到企业之间的紧密关系、交易规模, 甚至客户的核心需求与服务重点也能从中得到启发。

当数据量比较少时, 例如只有几百条交易记录, 可以直接通过数据可视化工具展示出来。当数据量特别大时, 往往需要进行大量的数据预处理和图计算, 将同属一个网络的节点和关系提前识别出来, 然后再通过可视化工具展示关系图。关于图计算和展示的可视化工具很多, 功能最强大的商业软件当属 IBM I2, SAS 也有专门的 SNA 功能模块。当然, 作为大数据领域的重要分析方法, 开源工具和代码自然是不少。本章介绍一款轻量级的开源软件——Gephi, 以帮助实现小数据量的 SNA 分析, 并结合前面介绍的网络识别算法来实现针对订单数据的关系网络构建和展示问题。应用篇将结合 Spark GraphX 和 Neo4j 建立关系网络查询系统, 实现大数据量的图计算和展示。

## 1. 订单数据处理

表 7-1 是一段时间内的订单明细数据(样例), 包括买家(采购商或消费者)、卖家(供应商)、商品、下单日期、订单金额及买家支付方式, 其中 YP 宝为电商自己的支付账户。

首先将买卖双方这段时间内相同商品的多笔交易进行合并, 以尽可能地缩减数据量, 如果买卖双方采用了多种支付方式, 则选择支付额最大的那种。汇总后的数据如表 7-2 所示。

表 7-1 一段时间内的订单明细数据(样例)

买家	卖家	商品	下单日期	订单金额 / 万元	买家支付方式
TH 汽车组件厂	江苏 BL 电源公司	电源	2015/2/9	200	I 银行
李四	DC-A 旗舰店	FC 新能源汽车	2015/2/1	13	YP 宝
李四	DC-A 旗舰店	FC 新能源汽车	2015/2/1	10	YP 宝
YQ-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	2015/2/16	44	C 银行
DS 汽车制造厂	苏州 NH 汽车座椅厂	座椅	2015/2/6	50	YP 宝
杭州 CZ 医院	DC-A 旗舰店	FC 新能源汽车	2015/2/3	100	C 银行
杭州 YM 车灯有限公司	YH 玻璃	玻璃	2015/3/1	40	YP 宝
张三	DC-A 旗舰店	FC 新能源汽车	2015/2/4	3	YP 宝
张三	DC-A 旗舰店	FC 新能源汽车	2015/2/5	10	YP 宝
DS 汽车制造厂	TH 汽车组件厂	发动机组	2015/2/4	200	YP 宝
DS 汽车制造厂	杭州 YM 车灯有限公司	车灯	2015/2/27	89	YP 宝
DS 汽车制造厂	HZ 汽车零部件公司	零部件	2015/3/1	25	YP 宝
DS 汽车制造厂	HZ 汽车零部件公司	零部件	2015/2/23	30	YP 宝



(续)

买家	卖家	商品	下单日期	订单金额 / 万元	买家支付方式
昆山 MQ 轮胎厂	无锡 SD 橡胶厂	橡胶	2015/2/18	12	Z 银行
DS 汽车制造厂	TH 汽车组件厂	发动机组	2015/3/2	200	YP 宝
DS 汽车制造厂	昆山 HP 汽车组件公司	变速箱	2015/2/25	100	YP 宝
TH 汽车组件厂	江苏 BL 电源公司	电源	2015/3/3	150	B 银行
DS 汽车制造厂	SD 汽车内饰厂	仪表盘	2015/2/7	68	YP 宝
杭州 YM 车灯有限公司	嘉兴 YF 照明	灯泡	2015/2/4	25	YP 宝
DC-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	2015/2/25	200	YP 宝
TH 汽车组件厂	江苏 JY 紧固件厂	紧固件	2015/2/1	24	I 银行
DS 汽车制造厂	苏州 NH 汽车座椅厂	座椅	2015/2/12	43	YP 宝
DS 汽车制造厂	苏州 GT 轮胎厂	轮胎	2015/2/17	79	YP 宝
昆山 HP 汽车组件公司	DY 特种钢有限公司	特种钢	2015/2/3	160	YP 宝
DS 汽车制造厂	TH 汽车组件厂	发动机组	2015/2/12	150	YP 宝
DS 汽车制造厂	昆山 HP 汽车组件公司	变速箱	2015/2/22	95	YP 宝
DS 汽车制造厂	昆山 MQ 轮胎厂	轮胎	2015/2/25	24	YP 宝
TH 汽车组件厂	江苏 BL 电源公司	电源	2015/2/3	150	I 银行
DC-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	2015/2/20	100	YP 宝
昆山 HP 汽车组件公司	DY 特种钢有限公司	特种钢	2015/2/2	100	YP 宝
DC-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	2015/2/4	150	YP 宝
昆山 HP 汽车组件公司	昆山 TS 装配工具厂	汽车工具套件	2015/2/27	10	YP 宝
昆山 HP 汽车组件公司	DY 特种钢有限公司	特种钢	2015/2/4	100	YP 宝
昆山 HP 汽车组件公司	慈溪市 X 汽车零部件厂	火花塞	2015/2/7	11	YP 宝
TH 汽车组件厂	浙江 YM 压铸公司	刹车片	2015/2/24	10	YP 宝
DS 汽车制造厂	苏州 NH 汽车座椅厂	座椅	2015/2/23	150	YP 宝
XJ 派出所	DC-A 旗舰店	FC 新能源汽车	2015/2/21	64	P 银行
杭州 XA 玩具厂	CH 塑料制品厂	车模型	2015/2/2	7	A 银行
杭州 XA 玩具厂	宁波北仑 DR 弹簧厂	片弹簧	2015/2/7	5	I 银行
王五	杭州 XA 玩具厂	玩家汽车	2015/3/15	0.5	YP 宝
...					

表 7-2 汇总后的订单数据 (样例)

买家	卖家	商品	订单金额	买家主要支付方式
DS 汽车制造厂	TH 汽车组件厂	发动机组	550	YP 宝
DS 汽车制造厂	昆山 HP 汽车组件公司	变速箱	195	YP 宝
DS 汽车制造厂	HZ 汽车零部件公司	零部件	55	YP 宝
DS 汽车制造厂	昆山 MQ 轮胎厂	轮胎	24	YP 宝
DS 汽车制造厂	苏州 NH 汽车座椅厂	座椅	243	YP 宝

(续)

买家	卖家	商品	订单金额	买家主要支付方式
DS 汽车制造厂	苏州 GT 轮胎厂	轮胎	79	YP 宝
DS 汽车制造厂	SD 汽车内饰厂	仪表盘	68	YP 宝
DS 汽车制造厂	杭州 YM 车灯有限公司	车灯	89	YP 宝
昆山 HP 汽车组件公司	慈溪市 X 汽车零部件厂	火花塞	11	YP 宝
昆山 HP 汽车组件公司	DY 特种钢有限公司	特种钢	360	YP 宝
昆山 HP 汽车组件公司	昆山 TS 装配工具厂	汽车工具套件	10	YP 宝
昆山 MQ 轮胎厂	无锡 SD 橡胶厂	橡胶	12	Z 银行
TH 汽车组件厂	浙江 YM 压铸公司	刹车片	10	I 银行
TH 汽车组件厂	江苏 BL 电源公司	电源	500	I 银行
TH 汽车组件厂	江苏 JY 紧固件厂	紧固件	24	I 银行
杭州 YM 车灯有限公司	YH 玻璃	玻璃	40	YP 宝
杭州 YM 车灯有限公司	嘉兴 YF 照明	灯泡	25	YP 宝
DC-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	450	YP 宝
YQ-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	44	C 银行
XJ 派出所	DC-A 旗舰店	FC 新能源汽车	64	P 银行
李四	DC-A 旗舰店	FC 新能源汽车	23	YP 宝
杭州 CZ 医院	DC-A 旗舰店	FC 新能源汽车	100	C 银行
张三	DC-A 旗舰店	FC 新能源汽车	13	YP 宝
杭州 XA 玩具厂	CH 塑料制品厂	车模型	7	A 银行
杭州 XA 玩具厂	宁波北仑 DR 弹簧厂	片弹簧	5	I 银行
王五	杭州 XA 玩具厂	玩家汽车	0.5	YP 宝
...				

## 2. 运行网络识别算法

指定要分析的买家或卖家，作为构建客户关系网络的初始点，可以是一个或者多个，本案例分析“DS 汽车制造厂”的关系网络。运行算法，限定输出两层关系，将得到“DS 汽车制造厂”的买卖信息及其买家卖家的买卖信息。根据 Gephi 的数据格式要求，数据至少包括三列：source（买家）、target（卖家）、商品及订单金额等反应客户关系的指标。因此，算法运行后的结果应该如表 7-3 所示。

表 7-3 客户关系网络数据

source	target	商品	订单金额	买家主要支付方式
DS 汽车制造厂	TH 汽车组件厂	发动机组	550	YP 宝
DS 汽车制造厂	昆山 HP 汽车组件公司	变速箱	195	YP 宝
DS 汽车制造厂	HZ 汽车零部件公司	零部件	55	YP 宝
DS 汽车制造厂	昆山 MQ 轮胎厂	轮胎	24	YP 宝

(续)

source	target	商品	订单金额	买家主要支付方式
DS 汽车制造厂	苏州 NH 汽车座椅厂	座椅	243	YP 宝
DS 汽车制造厂	苏州 GT 轮胎厂	轮胎	79	YP 宝
DS 汽车制造厂	SD 汽车内饰厂	仪表盘	68	YP 宝
DS 汽车制造厂	杭州 YM 车灯有限公司	车灯	89	YP 宝
昆山 HP 汽车组件公司	慈溪市 X 汽车零部件厂	火花塞	11	YP 宝
昆山 HP 汽车组件公司	DY 特种钢有限公司	特种钢	360	YP 宝
昆山 HP 汽车组件公司	昆山 TS 装配工具厂	汽车工具套件	10	YP 宝
昆山 MQ 轮胎厂	无锡 SD 橡胶厂	橡胶	12	Z 银行
TH 汽车组件厂	浙江 YM 压铸公司	刹车片	10	I 银行
TH 汽车组件厂	江苏 BL 电源公司	电源	500	I 银行
TH 汽车组件厂	江苏 JY 紧固件厂	紧固件	24	I 银行
杭州 YM 车灯有限公司	YH 玻璃	玻璃	40	YP 宝
杭州 YM 车灯有限公司	嘉兴 YF 照明	灯泡	25	YP 宝
DC-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	450	YP 宝
YQ-A 旗舰店	DS 汽车制造厂	FC 新能源汽车	44	C 银行
XJ 派出所	DC-A 旗舰店	FC 新能源汽车	64	P 银行
李四	DC-A 旗舰店	FC 新能源汽车	23	YP 宝
杭州 CZ 医院	DC-A 旗舰店	FC 新能源汽车	100	C 银行
张三	DC-A 旗舰店	FC 新能源汽车	13	YP 宝

### 3. 生成关系图

Gephi 主界面和数据导入选项如图 7-7 所示, 单击“输入电子表格”, 按照提示导入表 7-3 的数据 (csv 格式)。对于中文字符, 需要选择相应的编码格式, 这里为 GB18030。



图 7-7 Gephi 主界面和数据导入选项

Gephi 提供了多种算法, 这里我们采用 Yifan Hu 比例算法, 对算法感兴趣的读者可以参阅相关资料深入了解这个算法。

数据导入后, 如果未运行算法, 则如图 7-8 左侧的样子; 若运行算法, 则如图 7-8 右侧有规律的样子。

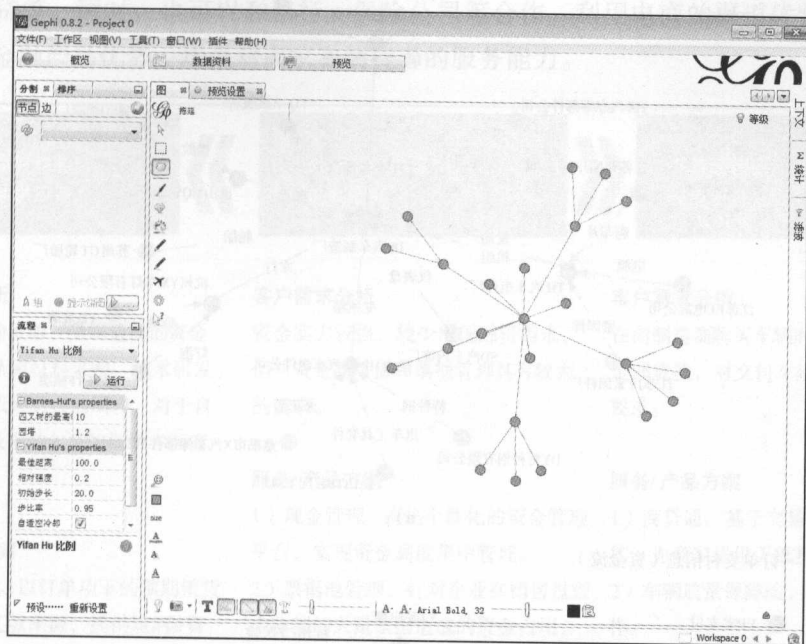


图 7-8 可视化: 由关系数据到关系图

Gephi 可提供图形修饰界面, 可以通过交互方式设置节点标签、连线标签、颜色、字体、节点和线条形状等参数, 可导出个性化的图形, 如图 7-9 所示。图 7-9a 为订单发货信息, 体现了物流方向与商品采购内容, 图 7-9b 为订单支付信息, 体现了资金流向。同时, 将经常使用银行账户支付的买家标识出来, 提供给客户关系管理部门, 进行 YP 宝支付促动。

### 7.3.3 分析结果的业务应用

#### 1. 帮助发掘新客户

通过 SNA 分析, 可以发现同处一个供应链中的商家及其上下游关系, 并通过可视化工具清晰地展示出一整条供应链 (或产业链), 告诉我们谁是上游原材料供应商、谁是下游采购商等信息。同时, 有了关系图的指引, 哪些商家经常使用电商自己的支付账户、哪些商家经常使用银行支付及支付的金额等都可以从图中获得。在客户关系管理中, 就可以按图索骥, 有选择性、针对性地开展营销和服务。



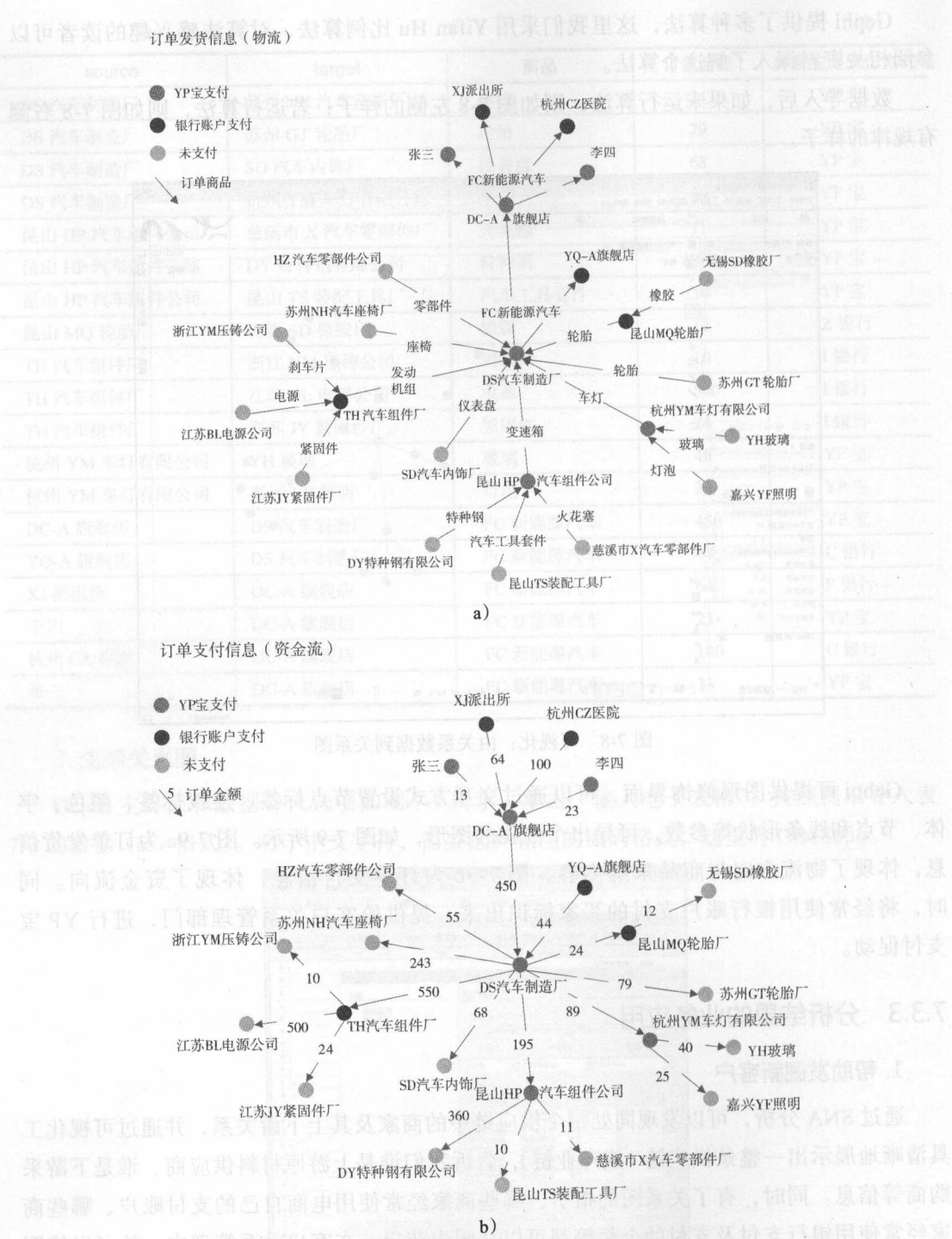


图 7-9 DS 汽车制造厂的上下游供应关系

## 2. 根据商家在供应链中的位置提供配套金融服务 / 产品

根据商家在供应链中的位置, 分析客户需求, 制定配套的金融服务方案, 如图 7-10 所示。处于核心位置的制造商, 资金实力较强, 可以提供余额增值服务; 上游原材料和零部件供应商, 往往会受到下游经销商压款, 可提供商户贷款; 针对终端消费者同样可提供小额消费信贷等。同时, 也可以和银行、保险公司等合作, 利用电商的渠道优势和银行与保险公司的金融产品优势, 增强对商家和消费者的服务能力。

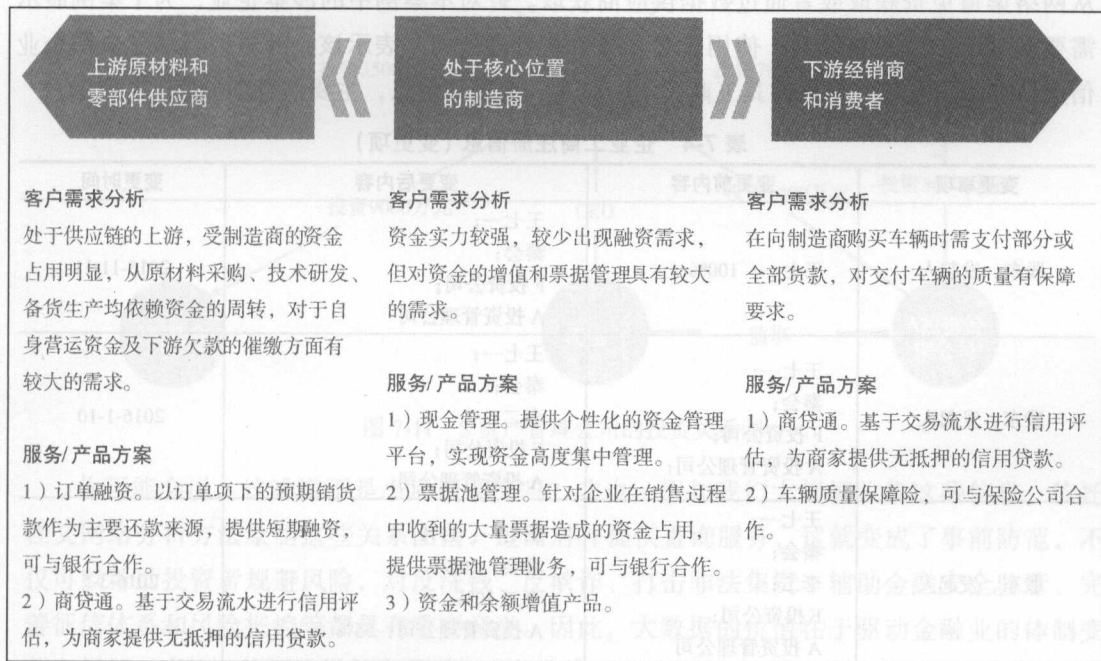


图 7-10 根据企业所处产业链位置提供配套金融服务

## 7.4 案例: P2P 投资风险防范

### 7.4.1 案例背景

近期, 某资产管理公司旗下 P2P 产品被查封, 其多名高管均以涉嫌非法吸收公众存款罪被检察机关批准逮捕。截至案发时, 该产品的累计交易金额超过 12 亿元, 注册会员数达 50 万人, 而拖欠投资者的赎回款总额已高达 7 亿元。

作为这几年异常红火的互联网借贷模式, P2P 正在进入风险暴露期, 一系列风险接踵而至, 例如诈骗、传销、自融、坏账多兑付难、恶意隐瞒事实、故意误导投资、监管政策不

确定等。那么如何及早发现存在问题的互联网借贷公司或 P2P 产品，并做到提前预警？业界研究人员总结了很多经验，从不同的角度出发识别风险点，也取得了一定的效果。笔者认为，大数据驱动的商业模式只有靠大数据才能做好风险防范。

## 7.4.2 防范方法

目前，工商注册、法院诉讼、税务、环保等各个社会领域的大数据都已开放使用，可从网络渠道免费获取或者通过数据供应商获取。针对本案例中的涉案企业，为了案例展示需要而又不涉及敏感信息，使用化名“S 资产管理公司”表示该企业，笔者从“全国企业信用信息公示系统”中查询其工商公示、股东、变更等信息，主要信息项如表 7-4 所示。

表 7-4 企业工商注册信息（变更项）

变更事项	变更前内容	变更后内容	变更时间
股东、发起人	王七一：100%	王七一； 秦会； F 投资公司； A 投资管理公司	2015-11-11
股东、发起人	王七一； 秦会； F 投资公司； A 投资管理公司；	王七一； 秦会； 李二； F 投资公司； A 投资管理公司	2016-1-10
股东、发起人	王七一； 秦会； 李二； F 投资公司； A 投资管理公司	王七一：15%； 秦会：20%； 李二 10%； A 投资管理公司：55%	2016-3-5
股东、发起人	王七一：15%； 秦会：20%； 李二 10%； A 投资管理公司：55%	NX 投资基金管理公司：50%； 王七一：10%； 秦会：15%； 李二：3%； A 投资管理公司：22%	2016-3-27
注册资本	5000 万元	10 000 万元	2016-3-27
...			

注：脱敏起见，本案例隐去真实名称，以化名展示。

将自然人和法人提取出来，建立如表 7-3 所示的格式，由于数据量较少，可以直接用 Gephi 绘制关系图，结果如图 7-11 所示。从图中可以看出，S 资产管理公司的 CEO 王七一一是 A 投资管理公司的监事，而 A 投资管理公司投资了 S 资产管理公司，自己投资自己！这是投资领域最大的禁忌，如果投资者早些看到这个关系图，显然就能够在投资前发现潜在的风险点，做到提前防范。

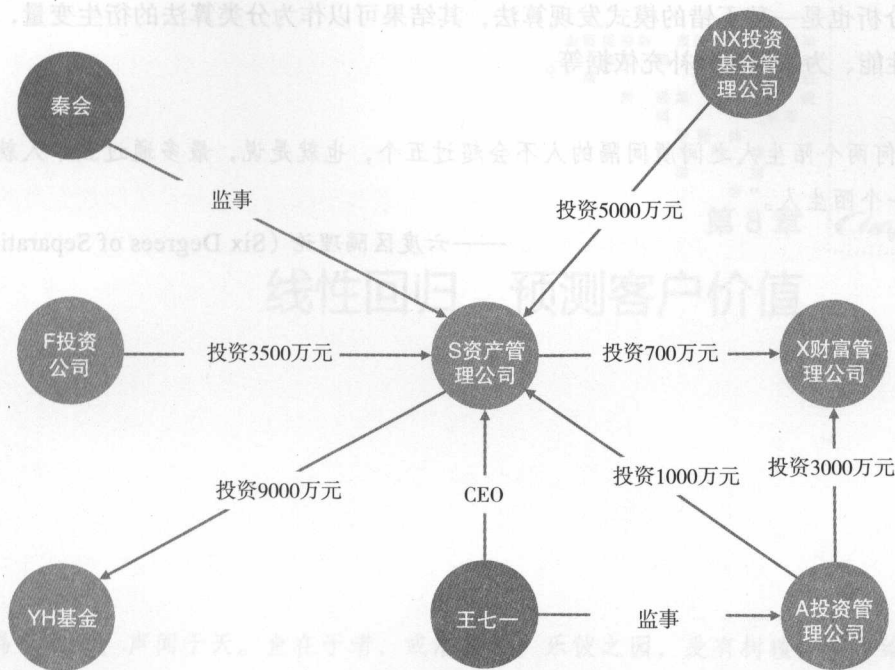


图 7-11 S 资产管理公司的投资关系图

你可能会说，这难道不是事后诸葛亮吗？非也，假如我们大规模收集这些信息，依托社交网络分析方法绘制这些关系图谱，进而对外提供查询服务，这就变成了事前防范，不仅可以帮助投资者规避风险，对反洗钱、反欺诈、打击非法集资、辅助金融安全监管、完善征信体系和风险把控等都具有重要意义。因此，大数据的价值在于驱动金融业的体制变革和创新。当然，使用这样的产品是可以收取费用的。

## 7.5 本章小结

随着 Facebook、Twitter、LinkedIn、微信等社交媒体的流行，对社交关系数据的挖掘成为近几年的一个热点话题，这些数据因为有了大量的互动而比传统的业务数据更加生动有趣，很多社交网络分析的结论也颠覆了我们的工作、生活和思维方式，这或许就是大数据时代所谓的“相关关系比因果关系更重要”的来由。本章应用社交网络分析构建客户上下游关系，发现潜在客户或者识别欺诈风险，帮助读者形象认识大数据关联分析的价值。两个案例仅作抛砖引玉，实际上社交网络分析的应用不应局限于此，社交网络本质是各实体间的关系，实体不局限于客户、账户等，可以是一些抽象的实体，例如文本挖掘中的语义关系，理论上说，所有多维表都可以用实体间的关系来展现。对于其他挖掘模型来说，社



“任何两个陌生人之间所间隔的人不会超过五个，也就是说，最多通过五个人就能够认识任何一个陌生人。”

## 线性回归，预测客户价值

鹤鸣于九皋，声闻于天。鱼在于渚，或潜在渊。乐彼之园，爰有树檀，其下维穀。它山之石，可以攻玉。

——《诗经·小雅·鹤鸣》

历史不会重演，但自有其韵律。

——马克·吐温

预测被称为大数据分析的圣杯，掌握预测技能也就成为众多数据分析人士的追求。预测分析似乎无所不能，把产品卖给需要的以及看起来并不需要的客户、在客户流失之前就发现征兆并将其留住、洞悉客户的潜在需求、帮助制定更英明的决策……但是，相比其他分析方法，预测分析是一项艰难的任务。成功的预测分析依赖于多个因素：①数据：数据越多，质量越高，与预测目标相关度越大、变量设计得越充分合理，预测的准确性和可靠度就越大。②预测分析软件：分析人员必须借助软件建立并评估模型，能够同时处理各种数据类型、计算高效软件让数据发挥作用。③执行力：再好的模型和规则，若无法植入应用，不能持续得到应用并优化，都只是花瓶。④精通预测的人：理解业务需求和目标，审视各方数据并设计有明确业务意义的变量。

预测分析方法有很多种，根据目标变量的类型，大体可分为两种：回归分析和分类。预测的结果如果是离散选项（比如用户会/不会购买某种产品、偏好短信/电话/微信渠道等），则通常称为分类；预测的结果如果是连续数值（比如交易量、客户价值等），则通常称为回归分析。本章将介绍统计学领域最经典的线性回归方法，随着对更多预测方法的了解，

你会发现其中可看到线性回归的影子。

## 8.1 数值预测

预测是数据挖掘的一项高级分析能力，技术要求高于统计、聚类、关联规则等其他分析的，数值预测尤其如此。按照 SAS 对数据分析能力的划分（见图 8-1），预测模型仅次于优化。



图 8-1 数据分析能力的八个等级（来源：SAS）

通常有以下两类数值预测问题。

1) 预测指标的走势, 比如销售收入、利润, 在未来一段时间内持续的变化状态, 属于图 8-1 中的“预报”类。

2) 预测每个个体的数值, 比如未来半年每个客户能够带来的利润, 属于图 8-1 中的“预测模型”类。

第一类预测通常使用时间序列分析方法, 应用线性回归解决的是第二类问题。然而, 对于个体的数值预测绝非易事, 下面举两个例子。

第一个例子, 银行客户价值分析的是预测每个客户未来可能带来的利润。有一位信用卡客户喜欢分期消费(贡献利息)、每月刷卡无数(贡献回佣)而且信用良好(风险低), 客户价值很高。有一天他跳槽到一家信用卡公司上班, 改用自己单位的信用卡, 原来的信用卡不再使用, 于是客户价值“跳崖式”下降。客户换工作几乎是无法预测到的, 其后果就是这些客户的价值预测全部失效。

第二个例子, 赌场要预测每个玩家未来半年的投注额, 以决定给他多大面额的 coupon, 有一位玩家历次的投注额都很大, 每次入住五星级宾馆、总统套房, 各项指标揭示这是一位出手阔绰的玩家。然而, 他有半年没有来……

因此, 虽然线性回归方法能够预测每个个体的具体数值, 但在实际应用时往往只针对群体。单就某些个体来说, 其关键影响因素可能我们并不掌握, 因此会有较大且无法解释的预测误差。

## 8.2 回归与拟合

### 8.2.1 回归就是拟合

在介绍线性回归的文章中, 有一个词的上镜率几乎达到百分之百, 这个词就是“拟合”。实际上, 线性回归就是线性拟合, 回归是对 regression 的直译, 拟合是意译。其含义为, 符合线性分布的点(用于建模的样本数据)都围绕在一条看不见的直线周围, 所有的点有向直线靠拢的趋势, 拟合就是要找到那条看不见的直线来替代这些点。由于直线可以轻松用公式描述, 以刻画目标变量  $y$  和众多  $x$  变量间的依赖关系, 通过  $x$  的已知值就可以估计或预测  $y$  值。

怎样找到那条直线呢? 所有的点到这条直线的距离加起来最小, 即残差平方和最小, 拟合曲线与散点分布最吻合, 这就是最小二乘法。

最后介绍线性这个概念。比如拟合每天学习时间和高考成绩, 可能就是线性的。若拟合收入高低和幸福指数, 很可能就不是了, 因为不是赚得越多越高兴, 可能到了很高的水



平，收入增加了很多，却幸福不起来，所以数据有可能是指数，有可能是二次函数，这些都归为非线性。后面介绍的 Logistic 回归就是这个思路。

8.2.2 在 Excel 中添加趋势线预测

经常应用 Excel 做数据分析的同学，肯定熟悉怎样使用趋势线进行预测。表 8-1 是一组客户的存款余额 (Deposit) 及半年的客户价值 (Value)，假设用存款余额 (x) 来预测客户价值 (y)。

表 8-1 客户的存款余额及半年的客户价值

Csr_id	Deposit(x)	Value(y)
1	5249	493
2	5569	511
3	5342	524
4	8575	813
5	9056	881
6	10 009	966
7	15 348	1524
8	19 087	1888
9	19 365	1892
10	19 772	1928
11	19 612	1932
12	23 583	2328
13	26 321	2595
14	26 197	2603
15	34 835	3461

第一步就是要分析两者是否有相关关系及是否有线性关系。对于样本量比较少的情况，除计算相关系数外，画散点图是更为直观的方法，如图 8-2 所示。

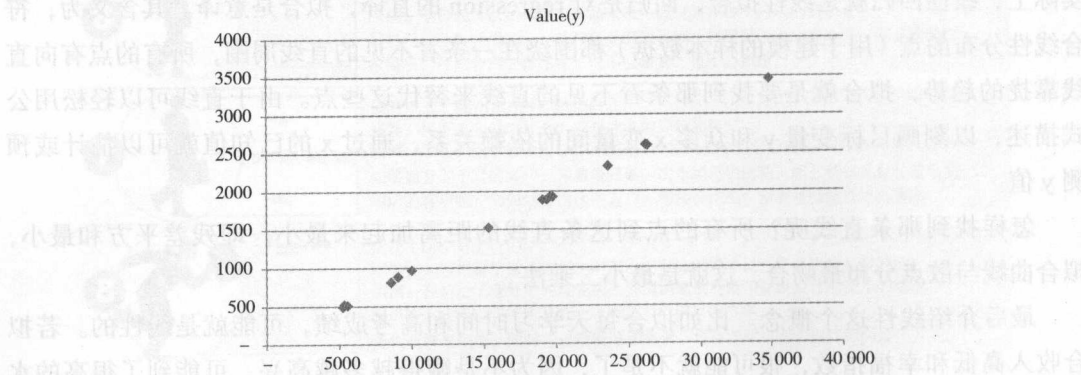


图 8-2 客户价值散点图 (横坐标 x: 存款余额)

从图 8-2 中可以看出, 散点大体沿着一条直线分布, 随着  $x$  的增加,  $y$  也在增加, 基本呈正相关关系 (如果计算相关系数, 值为 0.95)。但是,  $x$  增加 1 个单位,  $y$  增加多少, 也就是具体的量化关系还不知道。这时就需要添加趋势线了。

选中散点, 右键点选“添加趋势线”, 同时勾选“趋势线选项”, 如图 8-3 所示。

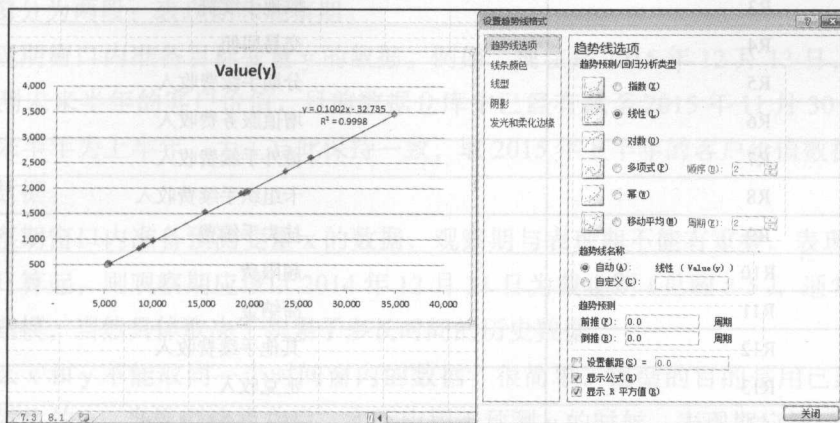


图 8-3 添加趋势线

通过添加趋势线, 得到  $x$  和  $y$  的量化关系:  $y=0.1002x-32.735$ ,  $R$  方为 0.9998。

如何得到这条趋势线的呢? 最小二乘法。所有的点到这条直线的距离加起来最小, 即残差平方和最小, 拟合曲线与散点分布最吻合。 $R$  方体现的就是吻合程度, 这里 99.98% 相吻合, 换句话说, 客户价值 ( $y$ ) 这个信息中有 99.98% 是可以通过存款余额 ( $x$ ) 解释的。例如某位客户, 存款余额为 10000, 那么他的客户价值就是 969, 从而实现了预测的功能。

**注意** 添加趋势线的拟合方法仅适用于单个预测变量的场景。

显然, 影响客户价值这个目标变量 ( $y$ ) 的因素有很多, 存款余额只是众多因素 (预测变量,  $x$ ) 中的一个, 最小二乘法仍然是拟合这条回归直线的基本方法, 但在这个由众多  $x$  构成的多维空间中, 就无法通过绘制散点图、添加趋势线来找到这条直线了。

## 8.3 案例：信用卡客户价值预测

下面以信用卡客户价值预测为例, 介绍整个线性回归模型的构建过程。

### 8.3.1 确定预测目标

首先, 确定预测的目标变量: 客户价值 ( $y$ )。对于每一个信用卡客户, 都有收入项  $R$

和成本项 C，客户价值 = 收入 - 成本。表 8-2 列出了比较典型的收入项和成本项。

表 8-2 信用卡客户的收入项和成本项

R1	利息收入
R2	年费
R3	取现手续费
R4	交易回佣
R5	分期手续费收入
R6	增值服务费收入
R7	境外手续费收入
R8	卡组织手续费收入
R9	挂失手续费
R10	超限费
R11	滞纳金
R12	其他手续费收入
R13	汇兑收入
C1	借入资金利息支出
C2	手续费支出
C3	其他营业支出
C4	营业税金及附加
C5	风险拨备
C6	客服成本
C7	催收反欺诈成本
C8	营销成本

客户价值预测，就是指在未来一段时间内每个客户能够带来多少利润，结果如图 8-4 所示。

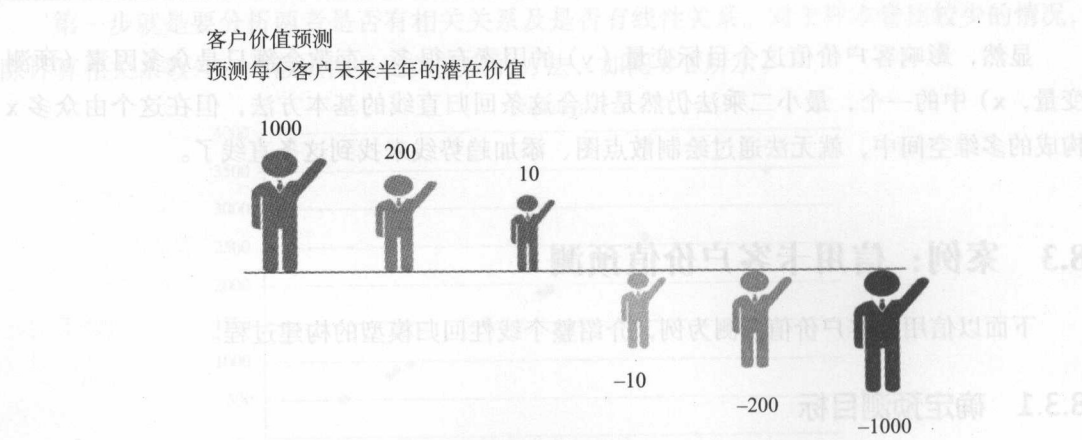


图 8-4 客户价值预测结果

### 8.3.2 准备建模数据

#### 1. 时间窗

时间窗, 是准备建模数据前必须理解的一个概念。

时间窗分为两段: 表现期和观察期。

在表现期窗口内准备目标变量  $y$  的数据。例如, 现在是 2015 年 12 月 13 日, 要建立一个模型预测未来半年的客户价值, 目前数据仓库中已经有截至 2015 年 11 月 30 日的数据, 考虑到未来半年为上半年, 为了与此保持一致, 取 2015 年上半年的客户价值数据作为目标变量  $y$  的数据。

在观察期窗口内准备预测变量  $x$  的数据, 观察期与表现期不能有重叠。表现期从 2015 年 1 月 1 日算起, 则观察期应该以 2014 年 12 月 31 日为截止期 (见图 8-5), 通常取 1 年的数据用于建模, 当然具体取决于积累了多长时间的历史数据。

为什么  $x$  和  $y$  不能取同一个时间窗内的数据? 很简单, 模型的目的是用已经发生的  $x$  预测尚未发生的  $y$ 。当模型建成之后, 实际应用于预测  $y$  的时候, 表现期应该还没有开始, 因此表现期内的  $x$  还没有发生, 假设  $x$  是表现期内的数据, 那么模型是没法使用的。

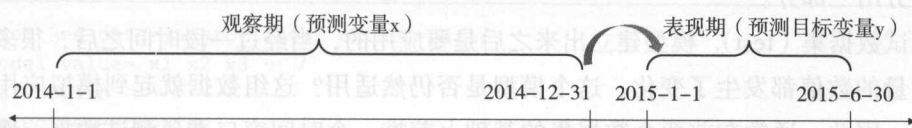


图 8-5 建模数据的时间窗示例

#### 2. 数据类型

另一个需要介绍的是数据类型, 通常有三类数据: 静态数据、时点数据和期间数据, 后两类是动态数据, 图 8-6 是几个示例。

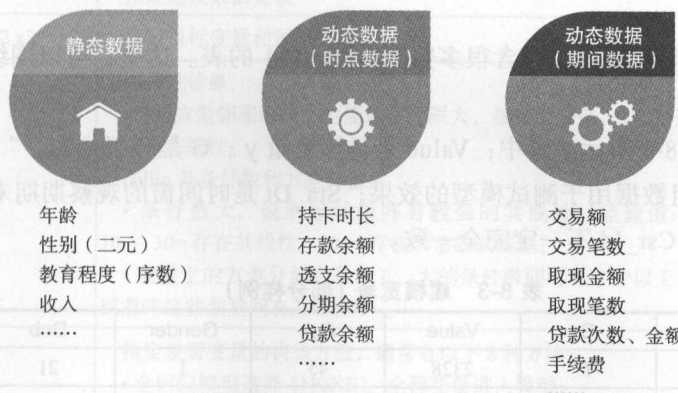


图 8-6 数据类型示例



静态数据是指在一段时期内不会发生变化的数据维度(变量),例如年龄、性别、教育程度、地域等。

时点数据是指截至某个时间点的数据维度,例如在表现期内截至每个月末的客户持卡时长、存款余额、透支余额等。时点数据是一种状态数据,通常余额类的维度都属于这类。

期间数据是指在特定时间段内的数据维度,例如客户在表现期内每个月的合计消费金额、合计消费笔数等,或者截至表现期末(2014/12/31)近3个月、近6个月、近1年的累计消费金额、累计消费笔数等。

当然,很多时候也会基于这三类数据设计一些新的数据维度,例如近1年每个月的消费笔数增长率、近1年的取现金额/消费金额。有时甚至是纯粹的数据维度的交叉计算,不关心实际含义。

### 3. 数据分组

建模的时候通常会对数据进行划分,用作不同的目的。SAS里一般的划分方法如下。

□ 训练数据集(train),用于训练模型,应用这组数据建模。

□ 验证数据集(validation),用于验证模型的稳定性,通常与训练集同一个来源,随机划分出一部分。

□ 测试数据集(test),模型建立出来之后是要应用的,当经过一段时间之后,很多预测变量的数值都发生了变化,这个模型是否仍然适用?这组数据就起到模拟应用的作用。因此,通常在前两个数据集的基础上前推一个时间窗口准备测试数据。例如每个月末运行的模型,准备测试数据的观察期和表现期就会整体向前移动一个月。

当然,对于样本量特别大的情景,例如,有数十万个甚至上千万个客户参与建模,随机划分出来的训练数据集和验证数据集会得出相同的模型,已起不到验证模型稳定性的作用。此时,测试数据集直接充当验证模型稳定性和适用性的角色。

### 4. 建模宽表

宽表就是横向无限扩展可以包含很多数据维度(列)的表。这里,可以围绕每个客户号后面添加各种特征。

最终的数据如表8-3所示。其中:Value是目标变量y;G是分组标志,“T”组数据用于训练模型,“V”组数据用于测试模型的效果;Sta\_Dt是时间窗的观察期期末,与G相对应;两组数据的客户Csr\_Id不一定完全一致。

表 8-3 建模宽表(部分样例)

Csr_ID	Sta_Dt	G	Value	Age	Gender	Dob	Deposit ...
1	2014/12/31	T	2328	45	1	21	95 435
2	2014/12/31	T	1932	49	1	12	77 778

(续)

Csr_ID	Sta_Dt	G	Value	Age	Gender	Dob	Deposit ...
3	2014/12/31	T	3461	43	2	9	73 890
4	2014/12/31	T	1888	56	2	108	73 379
1	2015/1/31	V	881	45	1	21	70 051
2	2015/1/31	V	813	49	1	12	67 675
3	2015/1/31	V	2603	43	2	9	54 506
4	2015/1/31	V	2595	56	2	108	54 446

8.3.3 模型拟合

1. 参数设置

如果不考虑变量的预处理，线性回归的建模过程主要包括共线性诊断和显著变量筛选两项内容。Proc Reg 是 SAS 中最基本的线性回归过程，如代码清单 8-1 所示，有关线性回归的各种需求都可以在这个过程中实现，这里主要介绍最常用的几个参数（见表 8-4）。

代码清单 8-1

```
Proc Reg data=datsrc_linear outset=para;
model value= x1 x2 x3 ... /
vif collin
selection=stepwise
sle=0.05 sls=0.05;
output out=Predfile p=Pred;
run;
```

表 8-4 Proc Reg 的参数含义

参 数	含 义
data=datsrc_linear	指定建模数据宽表
model value= x1 x2 x3 ... /	指定目标变量和预测变量
vif collin	共线性诊断 vif 是方差膨胀因子，可能趋于无限大，根据经验值，vif 大于 5 或 10 的变量与其他变量存有共线性。 collin 是条件数和方差分量： • 条件数大，说明设计矩阵有较强的共线性。经验值：0 ~ 10= 没有共线性；10 ~ 30= 存在共线性；30+= 存在严重的共线性； • 变量上的方差分量之和为 1，大的条件数同时有 2 个以上变量的方差分量超过 0.5，就意味这些变量间有共线性
selection=stepwise	指定显著变量的检验方法，通常有以下 8 种方法： • 全回归模型选择 (NONE)：全部变量进入模型；

(续)

参 数	含 义
selection=stepwise	<ul style="list-style-type: none"> <li>• 向前选择法 (FORWARD): F 检验的 p 值小于 sle 时, 选中变量, 逐个单向进入 (可能没有显著变量);</li> <li>• 向后消去法 (BACKWARD): p 值大于 sls 时, 变量剔除, 逐个单向剔除;</li> <li>• 逐步筛选法 (STEPWISE): 前向 + 后向, 前向选中一个变量后马上后向判断是否有变量不再显著, 最常用的检验方法;</li> <li>• 最大 R2 增量法 (MAXR)、最小 R2 增量法 (MINR): 先找到使 R2 最大的变量, 然后逐个比较 R2 增量 (总有变量进入模型);</li> <li>• R2 选择法、修正 R2 选择法 (ADJR SQ): 选择规定数量的变量数, 使 R2 最大, 最多 n-1 个</li> </ul>
sle=0.05 sls=0.05	<p>sle, 全称为 slentry, 规定变量进入模型的显著性水平, sle=0.05 表示 F 检验的 <math>P &lt; 0.05</math> 时变量可进入模型。</p> <p>sls, 全称为 slstay, 规定变量留在模型中的显著性水平, sls=0.05 表示 F 检验的 <math>P &lt; 0.05</math> 时变量可留在模型中</p>
outset=para	将通过显著性检验的变量、参数估计值和选择的统计量输出到指定的数据集 para 中, 这就是模型
output out=Predfile	把一些计算结果输出到指定的数据集 Predfile 中
p=Pred	<p>要求计算各观测点 (客户) 上目标变量的预测值, 命名为 Pred;</p> <p>* 将测试数据 “V” 组的目标变量 Value 全部置空, 则这部分客户将不参与模型训练, 但会被计算预测值, 正所谓建模和测试一举两得</p>

## 2. 结果解读

运行代码清单 8-1 后, 观察输出结果, 最主要的就是表 8-5 中的这两个表。

参数估计值表中, DEPOSIT 和 R6M\_AVG\_DEP 的方差膨胀因子大于 5, 存在轻微的共线性。而共线性诊断表中。条件指数为 6, 可以忽略。解决共线性的方法有很多种, 主成分分析、岭回归等方法消除共线性的技术手段; 而当表足够宽、变量特别多的时候, 通常采用另一种更为简单高效的方法, 即删除其中一个变量。当然, 如果显著变量特别多, 很多变量的方差膨胀因子大于 5 或 10, 则很难直接看出哪些变量存在共线性, 可以根据因子从大到小依次剔除。

t 值和  $\text{Pr} > |t|$  (P 值) 两列是变量的 t 检验和 F 检验的结果, P 值越小, 则 t 值越大, 变量的显著性水平越高, 此处设定显著性水平为 0.05, 因此最终留在模型里的变量的 P 都小于 0.05。当大部分变量的 P 值都小于 0.0001 时, 通过 t 值可以观察哪个变量更为显著。

通过参数估计值建立 y 和 x 的方程, 这就是模型, outset=para 输出的就是这些内容。这里,  $\text{VALUE} = 106.5 + 0.01384 \times \text{DEPOSIT} + 0.48447 \times \text{DOB} \dots$

掌握了任何客户的这组变量值, 就可以预测其客户价值了。

表 8-5 Proc Reg 的运行结果

参数估计值																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													
-------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

表 8-6 是客户价值的预测结果，在建模宽表的基础上增加了一列 Pred，是应用模型预测出的每个客户的客户价值。

表 8-6 客户价值的预测结果

Csr_ID	Sta_Dt	Age	Gender	Dob	Deposit	...	Pred
1	2015/2/28	45	1	21	95 435		2023
2	2015/2/28	49	1	12	77 778		2012
3	2015/2/28	43	2	9	73 890		3353
4	2015/2/28	56	2	108	73 379		1903

### 8.3.4 模型评估

模型构建出来后，拟合的好不好、是否可以投入应用，需要一些指标来评估。对于线性回归模型，主要有以下几个评估指标。

#### 1. R 方

R 方是留在模型里的预测变量与预测值之间的相关系数平方，体现 x 对 y 的解释能力。R 方介于 0 到 1 之间，越接近 1 表示模型拟合得越好，准备的数据维度中含有对 y 有显著影响的因素。

R 方可以作为选择不同模型的标准。如果在拟合数据之前，不能确定数据到底是什么模型，那么可以对变量的不同数学形式进行拟合，然后看 R 方的大小，R 方大的模型，说明这个模型对数据拟合得较好。

按照定义，模型中的变量数越多，则 R 方越大，但变量太多可能会带来过拟合或模型



不够稳定的问题，所以看一个模型的好坏，不仅要看 R 方，而且要看调整 R 方，后者对自由度（变量数）作了校正。

当然 R 方小也不一定意味着没有找到对 y 有显著影响的因素，可能是这些 x 与 y 是非线性关系，这时可通过数据预处理转换为线性关系。

2. 拟合曲线

观察模型效果最直观的方法是绘制散点图，对比目标变量的实际值和预测值，但当观测点（客户数）太多时，显然无法直接绘制这张图，可以浓缩一下。例如，按照预测值从高分到低分等分 100 组，计算每组的平均预测值和平均实际值，代替全部数据来绘制拟合曲线，如图 8-7 所示。应用“V”组测试数据绘制拟合曲线，而不是训练模型的“T”组数据，将更具说服力。

拟合曲线的解读方法：①两条曲线吻合，代表预测值与实际值一致；②实际值随预测值从高到低平滑下降，且下降幅度较大，代表模型能够较好地地区分出不同价值的客户。这个模型的拟合曲线就非常好。

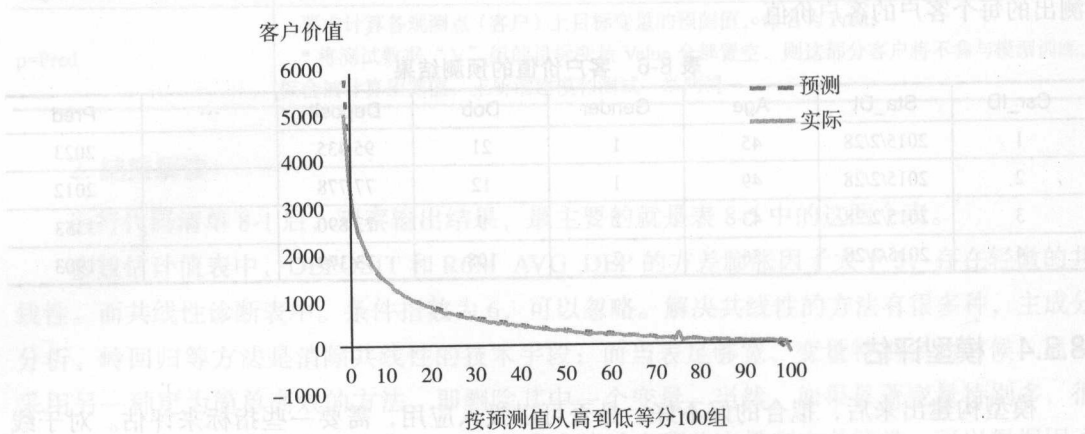


图 8-7 客户价值预测模型的拟合曲线

3. 预测误差

按照业务需求对客户分组，汇总每组客户的实际价值与预测价值，计算误差。表 8-7 是一个预测误差评估示例。可能会问：误差多少为好？越小越好。当然，一旦数据确定，变量设计完成，模型的效果实际也就大体确定了。

表 8-7 预测误差评估

客户分组	客户价值分组依据	人数	预测	实际	误差%
A	排名 TOP1%	63 760	314 078 203	357 121 389	-12%
B	排名 2% ~ 20%	1 211 450	1 849 879 852	2 059 872 871	-10%

(续)

客户分组	客户价值分组依据	人数	预测	实际	误差%
C	排名 21% ~ 50%	1 912 814	996 687 188	997 690 647	0%
D	排名 51% ~ 80%	1 912 815	405 237 919	373 163 353	9%
E	排名 80% ~ 95%	956 405	138 433 756	123 451 385	12%
F	排名 Lowest5%	318 803	57 360 433	35 348 584	62%

## 8.4 基于客户价值分层的业务策略

根据客户价值贡献的大小,对客户进行分级管理,是客户关系管理的核心策略。根据二八规律,现实当中 20% 的客户,带来 80% 的利润,而 80% 的客户只带来 20% 的利润。因而对客户按价值贡献大小分级管理,找出贡献最大的 20% 的客户,对他们提供区别于其他客户的服务,有助于提高他们的忠诚度,从而实现相同的投入带来最大的回报。如何找到价值贡献最大的 20% 的客户呢?很简单,只要将客户的贡献(销售额或利润)进行统计,按二八规律进行分割即可。

目前比较流行的一种客户分级方式,叫金字塔客户分级模型,已在第 5 章提到(见图 5-3),该模型将客户分为钻石客户、黄金客户、白银客户、普通客户和垃圾客户,对客户建立一种金字塔式的管理方式。当然,也可以根据自己的实际情况来划分客户级别。客户分级管理除了能知道对自己价值贡献最大的客户是谁,并分别对待以获取他们的忠诚,还能据此指导我们的业务拓展,既对贡献最大的 20% 的客户进行分析,提取他们的共同特征,然后以此指导销售工作,对与符合该特征的潜在客户进行特别的关注。

根据对客户价值的预测,按照金字塔理论可将全部信用卡客户划分为三层,分别提供差异化的服务、产品和营销策略(见图 8-8),集中服务于高价值客户群体,达到优化利用有限资源的目的。

## 8.5 本章小结

线性回归分析本身并不困难,除 SAS 外,几乎所有的分析软件都提供了这个模型的建模功能,且算法大同小异,分析人员只要知道如何解读几个关键参数即可,并不需要从头开发算法。要成为高手,取决于数据预处理的能力。

线性回归模型只能处理连续型的变量,但不意味着离散型的变量就要丢弃,一般采用以下两种方法。

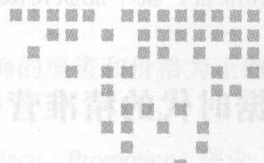
	高价值客户	中等价值且敏感客户	低价值客户
电话接听差异化策略	1. 优先转接、优先转给资深坐席； 2. 放宽对客户投诉受理需求； 3. 客户提出不愿收到营销信息时，1个工作日内完成屏蔽； 4. 对于投诉，一通电话解决客户问题	1~3：同前； 4. 4小时内处理敏感客户投诉事务，3日内解决，并在不打扰客户的前提下，每日向其更新投诉解决进程	1~3：同前； 4. 对于普通投诉，要求一通电话解决客户问题；对于重要投诉且投诉问题涉及多个部门、供应商的，要求3个工作日内予以解决
催收差异化策略	1. 在催收初级阶段弱化催收强度，以提醒为主，减少对客户的干扰 2. 电催、信函及短信话术亦被专门制定，用语礼貌委婉 3. 体现为客户用卡安全、享受优质服务提供保障	按正常流程处理	1. 按正常流程处理 2. 在催收、信息修复、外部催收、资产重组、交易监控等环节根据客户价值细分客户群，采取差异化的服务
营销差异化策略	1. 同类活动，每个客户只收到一条信息 2. 循环信用、分期和取现活动各渠道告知均不达到客户 3. 各类需要注册的活动默认注册	1~2：同前 3. 屏蔽所有的电话外拨销售	1. 以增加收益的活动为主，如分期、取现等。其他活动屏蔽 2. 销卡不挽留

图 8-8 基于客户价值分层的差异化策略

1) 通过其他分析方法筛选出显著的离散变量，划分客群，针对每个客群分别建立线性回归模型。

2) 离散变量连续化，将变量的每个选项转换为 0/1 二元变量或计算 WOE 值替代原变量（参阅数据预处理章节）。

建模过程中尤其值得注意的是避免落入因果倒置的陷阱，一定  $x$  是因， $y$  是果，不能颠倒。笔者此前已碰到过很多案例，甚至发生在有多年分析经验的人员身上，例如“分析发现，高价值信贷客户的 A 类融资产品持有率非常高，因此认为高价值信贷客户偏好 A 类融资产品，应该针对这些客户推荐 A 类融资产品”，而实际上信贷客户的价值根据产品的利息贡献计算，A 类融资产品的利率要高于其他产品，因此不是高价值客户喜欢 A 类融资产品，而是因为签约了 A 类融资客户价值才更高。通过时间窗将  $x$  和  $y$  的发生顺序区分开来， $x$  发生在先， $y$  发生在后，可在很大程度上避免因果倒置，但对于模型的显著性变量仍需要认真分析对预测真正起作用的原因是什么，以让模型站得住脚。



## 第9章 Chapter 9

# Logistic 回归，精准营销的主要支撑算法

凡事豫则立，不豫则废。言前定则不跲，事前定则不困，行前定则不疚，道前定则不穷。

——《礼记·中庸》

虽然人与人之间有很多不同，但我们的可预测程度都差不多，无情的统计规律使得异类根本不存在。人类行为 93% 是可以预测的。

——巴拉巴西《爆发》

人类个体的行为看起来是随机的，每个事件的发生都是小概率事件，谁会青睐汽车公司的新款车型、谁将拖欠银行的贷款、谁将光临某个餐馆，似乎很难预知。但很难不等于不能，只要有足够多的个体，每个个体在事情发生之前有足够多的历史数据，就可以通过大量的归纳、推理，计算出事情发生前的必然规律。当然，人脑无法承载这么大规模的计算，所以人难以预测，但机器可以，只要机器拥有足够多的数据和学习算法，就能够实现预测，100% 是神话，93% 却是可能的（巴拉巴西在其《爆发》一书中就抛出人类行为 93% 可以预测的言论）。

除了足够多的数据，机器学习算法是另一个必要条件。机器学习算法有很多，目前主流的莫过于分类算法，与专注于连续值预测的回归类算法不同，分类算法通过对类别（主要是二元，例如购买 / 不购买、拖欠 / 不拖欠）的对比，发现其中的统计规律并用于预测。Logistic 回归算法因其操作简单、结果易用、拓展性强等优点而得到广泛应用。

这样，我们就能够预知谁会青睐汽车公司的新款车型、谁将拖欠银行的贷款、谁将光临某个餐馆，让商业决策有的放矢，获得成功。正如毛泽东在《论持久战》中提出的：“没



有事先的计划和准备，就不能获得战争的胜利。”

## 9.1 大数据时代的精准营销

### 9.1.1 精准营销

精准营销并不是一个新概念，而大数据却让精准营销焕发出新的生机，想想看过的大数据应用案例，精准营销占据多少比例。如果大数据是一种解决问题的能力，为商业创新提供了更多可能，那么精准营销究其本质是一种营销理念，通过对关键营销要素的把握，达到“予营销于无形、变营销为服务”的终极目标。

精准营销的关键词是“精”和“准”。所谓少而精，与传统广而告之、撒网式营销模式相比，精准营销尽可能地将每次活动的受众客户群控制在一个比较小的范围，以求更高的投入回报和不断优化的营销过程，前提则是“准”。以前写作文的时候，老师经常强调六要素：时间，年月日叙述清楚；地点，事件发生的环境场所；人物，不写人物就成了无头案；原因，为何发生这件事；经过，理清事件来龙去脉；结果，交代清楚结局（在这一点上网络小说尤其重视）。精准营销比较类似，通常也是对六个要素的准确衡量，从而达到“准”的目的，如图 9-1 所示。



图 9-1 精准营销的六要素

- 1) 人是第一要素，需要准确圈定本次营销活动或信息推送的目标客户。
- 2) 时间，与客户接触或信息送达客户的时间点要拿捏好。
- 3) 地点包括两个方面：客户经常活动的范围；合适的营销场景。
- 4) 产品是营销的内容，基于对客户偏好的准确分析，提供客户感兴趣的商品、信息或服务。
- 5) 渠道，以什么方式与客户接触、投递信息。

6) 价格, 这个要素比较宽泛, 例如比较优惠的商品价格、给予目标客户有吸引力的支付折扣、客户可以选择的优惠项等。

一句话, 精准营销就是在正确的时间与地点以正确的渠道和价格为正确的人提供正确的产品。

其理论基础无论是强调产品的 4P (Product, Price, Place, Promotion) 理论, 还是强调消费者的 4C (Customer Value, Customer Communication, Customer Cost, Customer Convenience) 理论, 营销的重点都是从“请消费者注意”转变为“请注意消费者”, 为商业银行创新营销模式拓展了思路, 具体是实现三个转型: 其一, 单客户营销/服务向客户群营销/服务模式转变; 其二, 单产品销售向全交易流程的产品组合销售转变; 其三, 分销渠道、物理渠道重心向数字化渠道转移。关于大数据下的精准营销与传统营销的差异, 如图 9-2 所示。

	传统营销模式	大数据营销模式
① 营销模式	普惠式、大概率	目标客户精准营销
② 营销手段	经验依赖	数据依据
③ 营销观念	被动等待客户上门	主动发现并满足客户需求
④ 驱动因素	产品	客户需求
⑤ 客户响应率	低	高
⑥ 营销执行	一次性	营销+评估, 滚动
⑦ 每次营销人数	全量、万	千
⑧ 活动频度	不定期	日常
⑨ 营销风格	单一产品	产品组合

图 9-2 精准营销与传统营销的差异

### 9.1.2 基于大数据的精准营销模式

有了大数据做引擎, 精准营销得到了极致演绎, 营销模式不断创新, 应用场景越来越细分。归纳起来, 四类模式最具代表性, 如图 9-3 所示。



图 9-3 典型的大数据营销模式

### 1. 清单筛选式营销

数据仓库+模型筛选+目标客户清单,目前业界最常见的大数据营销手段,通过模型筛选目标客户群,在营销服务和业务处理环节主动推送产品/服务信息。

### 2. 模型触发式营销

客户条件库+模型筛选+事件触发,多用于接触点式营销。其一,借助客户到网点/自助设备办理业务、来电咨询、POS消费等时机,发起服务或信息;其二,制定某类客户或产品的服务方案,当客户满足某个条件或达到某标准时发起服务或信息。

### 3. 事件式营销

以“事件”作为营销活动的触发点,当客户行为变化、发生特定交易时,触发信息推送或跟进营销服务。

### 4. 大数据预测营销

潜力挖掘和事前营销,更具前瞻性,在广泛搜集客户信息的基础上,预测某类客户的潜在需求,据此设计、研发和销售相应的产品。

## 9.1.3 如何做到精准

要让营销做到精准,就需要准确预测这些营销要素,例如,一组客户对特定产品的潜在需求、对各种接触渠道的喜好等,基于预测结果缩小目标客户的范围,只筛选更可能购买该产品的客户做营销,采用客户最喜欢的接触渠道推送营销信息。

这正是 Logistic 回归模型所擅长的。

线性回归模型擅长预测数值,目标变量也是数值;Logistic 回归模型则擅长预测一种状态或者事件发生的概率,目标变量是类型,例如客户是“买”还是“没买”某个产品,客户喜欢的接触渠道是电话、网络还是面对面,这类问题统称为“分类”,Logistic 回归和决策树是解决分类问题的高手(实际上大部分预测模型的本质都是分类,在后面建模部分详述)。

当产品营销结束进入历史数据时,每个客户的“购买状态”只能是唯一的,客户要么买了,要么没买,是一个二元变量(0或1),但在营销之前这个状态是介于买与没买之间的,购买的可能有多大,在统计学上表示可能的概率,所以 Logistic 回归预测的结果是某个状态发生的概率(见图 9-4),客户可能购买某产品的概率是多少、客户接受各个渠道的概率是多少,基于预测概率来确定营销关键要素,营销得以精准。



图 9-4 Logistic 回归模型的预测结果

## 9.2 Logistic 回归算法介绍

### 9.2.1 算法原理

Logistic 回归是当前业界比较常用的机器学习算法, 用于估计某件事情的可能性。所谓机器学习就是让计算机模仿人的思维模式去获得知识、得到结论, 学习的依据是数据。这里虽然是介绍原理, 但并不想用公式推导的方式来介绍。

这里以介绍正常人的学习过程为例, 表 9-1 是一组天气数据样例。已知 10 月 12 日 ~ 14 日每天的空气湿度、平均温度、昼夜温差、风速及明天阴晴等气象数据, 又知 10 月 15 日的天气情况, 问 10 ~ 16 日是阴天还是晴天? 正常人的分析过程大概是这样: 14 日是晴天, 12 日、13 日是阴天, 14 日的空气湿度、平均气温、昼夜温差、风速等都与 12 日、13 日有很大差异, 空气湿度和平均温度更低, 昼夜温差和风速更大……15 日的这些指标与 14 日的比较接近, 与 12 日、13 日的差别较大, 经过归纳推理, 16 日应该与 15 日一样, 是晴天。

表 9-1 天气数据样例

日期	空气湿度	平均温度	昼夜温差	风速	...	明天阴晴
10 月 12 日	90	30	5	2		阴
10 月 13 日	85	29	6	3		阴
10 月 14 日	60	27	10	5		晴
10 月 15 日	59	26	11	4		?

机器学习的过程只是使用一种量化的方法。首先将历史数据中的天气阴晴情况 (预测目标,  $y$ ) 分为阴天和晴天两类; 然后对比两类天气的各项气象数据, 既然天气阴晴与多个因素 (自变量,  $x$ ) 有关, 那么要分别对比晴天和阴天时各个因素的差异, 差异大, 说明与阴晴相关, 是显著变量, 差异小, 则说明与阴晴不相关。于是得到一组与  $y$  相关的显著变量  $x_1, x_2, \dots, x_m$ , 那如何考量这些变量呢? 对应晴天, 有的值大, 有的值小, 求和是不行的。机器学习算法会量化评价每个变量与  $y$  的关系并给出打分, 同时也会对比各个变量的重要



性, 分配权重, 用加权和判断天气阴晴。加权和是从负无穷到正无穷的数值, 这样看起来和线性回归的输出相同。是的, 实际上, Logistic 回归就是对预测目标先做 logit 变换, 进行线性回归得到类似加权和的输出, 然后将加权和压缩到 0 和 1 之间来表达“可能性”。可以说, Logistic 回归本质上就是一个被 Logistic 方程规格化后的线性回归。

## 9.2.2 关键步骤

下面再用一个例子介绍 Logistic 是怎么做的以及 logit 变换是怎么回事。表 9-2 是一组客户的交易活跃度及其流失情况, 假设交易活跃度是反映信用卡客户流失的一个因素 ( $x$ ), 这是一个值为 1 ~ 9 间的顺序变量, 那么如何通过一个客户的交易活跃度预测其流失概率?

表 9-2 客户的交易活跃度及其流失情况

交易活跃度 ( $x$ )	客户数量 / 人	流失人数 / 人	占比 / % (流失率 $p$ )
1	910	35	4%
2	1165	135	12%
3	1120	220	20%
4	1180	455	39%
5	1125	650	58%
6	1075	840	78%
7	1105	970	88%
8	1000	955	96%
9	1320	1,300	98%

### 1. 建立交易活跃度与流失率的关系

模型是对大部分客户共性规律的总结, 不计较单个客户, 因此先按照  $x$  的取值合并客户并计算流失客户占比。 $x=1$  的客户共 910 人, 其中 35 人流失, 占 4%, 依此类推, 交易活跃度与流失率 (预测目标,  $p$ ) 的总体关系是一条近似 S 形状的曲线, 如图 9-5a 所示, 两头平缓中间陡峭。 $x$  与  $p$  非单调的线性关系, 无法用类似  $p=a+bx+\dots$  的公式描述。

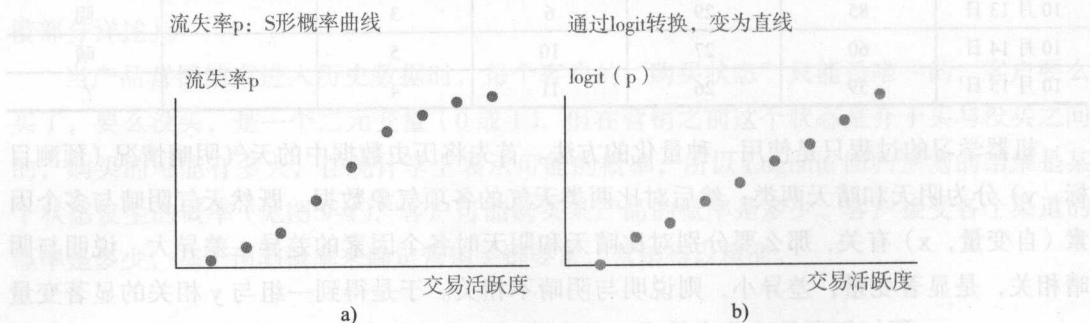


图 9-5 交易活跃度与流失率的关系曲线

## 2. logit 变换

logit 变换的唯一目的就是将在 S 曲线转换为近似直线的关系, 方法是取  $p/(1-p)$  的自然对数,  $y=\text{logit}(p)=\ln(p/(1-p))$ , 方法如表 9-3 所示。变换后  $y$  与  $x$  的关系如图 9-5b 所示。

表 9-3 客户的交易活跃度及其流失情况

交易活跃度 (x)	客户数量 / 人	流失人数 / 人	占比 / % (流失率 p)	1-p	$p/(1-p)$	$\ln(p/(1-p)) = y$
1	910	35	4%	96%	0.04	-3.22
2	1165	135	12%	88%	0.13	-2.03
3	1120	220	20%	80%	0.24	-1.41
4	1180	455	39%	61%	0.63	-0.47
5	1125	650	58%	42%	1.37	0.31
6	1,075	840	78%	22%	3.57	1.27
7	1105	970	88%	12%	7.19	1.97
8	1000	955	96%	5%	21.22	3.06
9	1320	1300	98%	2%	65.00	4.17
合计	10 000	5560	56%	44%	1.25	0.22

## 3. 线性回归

这样, 就可以用线性回归方法拟合曲线, 得到  $y$  与  $x$  的函数关系, 即  $y=0.8889x-4.0376$ ,  $R^2=0.997$ 。  $y$  是一个连续数值, 并非介于  $(0, 1)$  的概率值。

## 4. 预测并进行反 logit 变换, 得到流失概率

对于任何一个客户, 只要知道了交易活跃度  $x$ , 就可以根据公式得到加权和  $y'$  值, 例如  $x=1$ , 则  $y'=-3.149$ 。对  $y'$  压缩到  $(0, 1)$  之间的方法是反 logit 变换  $p'=ey'/(1+ey')$ , 这样就预测了客户的流失概率。计算方法如表 9-4 所示。

表 9-4 客户流失概率预测

交易活跃度 (x)	占比 / % (p)	$Y=\ln(p/(1-p))$	$y'$ (预测)	$ey'$	$1+ey'$	$p'$ (预测)
1	4%	3.22	-3.15	0.04	1.04	4%
2	12%	-2.03	-2.26	0.10	1.10	9%
3	20%	-1.41	-1.37	0.25	1.25	20%
4	39%	-0.47	-0.48	0.62	1.62	38%
5	58%	0.31	0.41	1.50	2.50	60%
6	78%	1.27	1.30	3.65	4.65	79%
7	88%	1.97	2.18	8.89	9.89	90%
8	96%	3.06	3.07	21.62	22.62	96%
9	98%	4.17	3.96	52.59	53.59	98%

## 9.3 案例：信用卡消费信贷产品的精准营销

精准营销只是 Logistic 回归的一个应用领域，实际上，Logistic 回归几乎可以应用于任何常见的业务领域，因为分类问题实在是无处不在，例如在风险防控领域中甄别高风险案件、在反欺诈领域中分析欺诈规则、在客户画像中帮助筛选与业务背景强相关的信息等。这里我们仍然以精准营销案例介绍如何应用 Logistic 回归，原因很简单，营销在各个行业是共通的，而且直观易懂。

### 9.3.1 案例背景

消费信贷是区别于传统的抵押、质押贷款的一种新型贷款产品，依据对客户信用的评分发放相应额度的贷款，具有审批速度快、无需抵押或担保的特征，非常适合那些短时间内急需小额资金周转而又无法提供抵押物或担保的消费者。随着大数据征信技术的越来越完善，消费信贷已成为近几年最主要的互联网金融创新产品。除各类互联网金融公司外，商业银行也顺势而为，纷纷推出形态各异并实质相似的消费信贷产品，其中尤以信用卡消费贷款为主。信用卡消费贷款是以信用卡持卡人为发放对象的现金分期，随着信用卡普及率的提高，信用卡消费贷款成为不少银行主推的业务。根据波士顿咨询报告，消费信贷也是中国零售银行创新突围的三大热点之一（另两个热点是数字化金融和社区银行）。

对于这类新兴产品，消费者认知与偏好分化。芸芸众生，如何判别出具有足够接受度的客户？况且客户对信贷产品的需求本质上是对短期资金的需要，时效性非常强，营销机会稍纵即逝，这就与一些生活中可有可无产品的营销不同。实际上，客户大量的消费行为、生活形态以及第三方数据中揭示出的资金需求，这些数据也是各家银行的业务系统每时每刻都在产生的。预测资金需求的基础是先按照客户整合所有体现资金需求的信息，在唯一识别客户的 ID 后面一一加工出若干数据变量（参阅第 4 章），并进一步对这些数据进行分类和标签化，建立客户视图。标签化的目的是从定量数据中归纳出定性的特征，将复杂的数据简单化，便于理解和应用，具体内容可参阅第 12 章。对于耳熟能详的 360 度客户视图发表点看法，实际上根本就不存在 360 度的客户视图，任何企业掌握的信息只是客户完整信息拼图中的几块而已，引进各方数据尽可能地多拼接几块、更全面地了解客户，是大数据时代企业需要研究的重要课题。

接下来以某银行信用卡公司的一款消费信贷产品“任意贷”为例，介绍如何应用 Logistic 算法建立响应率模型，并帮助精准营销。

### 9.3.2 数据准备

对于响应率模型，首先是确定预测目标，然后分析可能相关的因素，设计预测变量。



不同的业务场景, 例如产品的目标客户定位、客户流失预警、客户信用评分、风险评分等, 建模的差异体现在目标变量的定义, 数据准备与建模过程基本相同。

### 1. 确定目标变量和预测变量的时间窗

假设建模的时间为 2015 年 3 月, 以后将应用模型预测一个月内客户申请“任意贷”的概率。选取 2015 年 2 月作为表现期, 根据该期间内客户有没有申请“任意贷”及类似的消费信贷来定义目标变量  $y$ , 申请, 则  $y=1$  (响应), 否则  $y=0$  (未响应); 以 2014 年 2 月至 2015 年 1 月作为观察期, 采用这段时间内的客户数据设计预测变量  $x$ 。这就是时间窗的概念,  $x$  和  $y$  的取值窗口完全错开,  $x$  在前,  $y$  在后, 如图 9-6 所示。当然, 表现期取多长时间取决于业务需求, 时间越短, 预测的时效性越高, 但准确性会下降; 而观察期取多久则完全取决于积累了多长时间的历史数据, 但时间太长也不推荐, 一则太久远的数据与目标变量的关系不会太大, 二则增加计算成本, 一年是一个比较合适的周期。

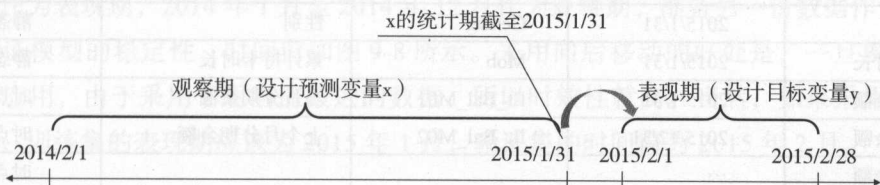


图 9-6 建模数据的时间窗

### 2. 设计预测变量

有人会问, 业务系统和数据仓库中有那么多数据, 应该使用哪些来建模呢? 答案是看你的计算能力。在离线环境中建模, 变量多多益善, 原因很简单, 谁也不知道哪些因素能反映客户的资金需求, 更多变量意味着更准确预测的可能。当然, 如果已经建立了完善的客户视图, 将与客户相关的特征都按照客户号一列一列地加工成变量, 那么在准备建模数据的时候就方便多了。图 9-7 是一个较为常见的信用卡客户视图示例, 共 7 个维度、数千个特征变量。

如果没有客户视图, 那么可参照第 4 章介绍的变量衍生方法设计预测变量。这里再介绍一个比较容易理解的变量设计方法: 客户特征  $\times$  统计期。对于年龄、性别、持卡时长等静态变量 (在整个观察期内基本不变), 直接取观察期末 2015/1/31 当天的数据; 对于分期余额等余额类变量, 属于时点型动态数据, 从 2015/1/31 依次向前取每个月月末的状态数据; 对于消费金额等发生额类变量, 属于期间型动态数据, 从 2015/1/31 依次向前取每个月、近 3 个月、近 6 个月以及全年的汇总数据。表 9-5 是一个变量设计示例。



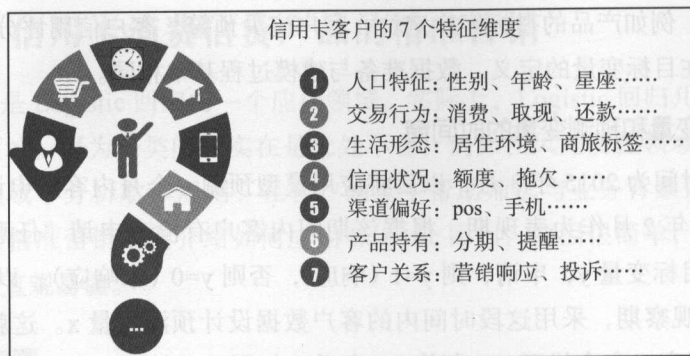


图 9-7 信用卡客户视图的 7 个特征维度

表 9-5 客户特征 × 统计期变量设计示例

客户特征	统计期	预测变量 (英文名)	预测变量 (中文名)	变量类型
年龄	2015/1/31	Age	年龄	静态
性别	2015/1/31	Gender	性别	静态
持卡时长	2015/1/31	Mob	累计持卡时长	静态
分期余额	2015/1/31	Ilt_Bal_M01	当月分期余额	时点
分期余额	2015/12/31	Ilt_Bal_M02	上个月分期余额	时点
分期余额	...			时点
分期余额	2014/2/1	Ilt_Bal_M12	观察期首月分期余额	时点
消费金额	2015/1	Cns_Amt_M01	当月消费金额	期间
消费金额	2014/12	Cns_Amt_M02	上个月消费金额	期间
消费金额	...			期间
消费金额	2014/2	Cns_Amt_M12	观察期首月消费金额	期间
消费金额	14/11 ~ 15/1	R3M_Cns_Amt	近 3 个月消费金额	期间
消费金额	14/8 ~ 15/1	R6M_Cns_Amt	近 6 个月消费金额	期间
消费金额	14/2 ~ 15/1	R1Y_Cns_Amt	近 12 个月消费金额	期间

基于这些基础变量，可以计算在表现期的走势、均值、方差等统计量，或者变量间进行交叉计算，生成新的衍生变量，例如近一年消费金额的复合增长率、单月最多来电次数、笔均消费金额波动区间等，从更多的角度体现客户行为。表 9-6 是一个衍生变量示例。

表 9-6 衍生变量示例

衍生变量英文名	设计方法
R1m_Avg_Tot_Lmt_Utl	当月总额度使用率
R3m_Avg_Tot_Lmt_Utl	近 3 个月平均总额度使用率
R6m_Avg_Tot_Lmt_Utl	近 6 个月平均总额度使用率
R12m_Avg_Tot_Lmt_Utl	近 12 个月平均总额度使用率
R1m_Avg_Csh_Tot_Lmt_Utl	当月取现占总额度使用率

(续)

衍生变量英文名	设计方法
R3m_Avg_Csh_Tot_Lmt_Utl	近 3 个月平均取现占总额度使用率
R6m_Avg_Csh_Tot_Lmt_Utl	近 6 个月平均取现占总额度使用率
R12m_Avg_Csh_Tot_Lmt_Utl	近 12 个月平均取现占总额度使用率
R1m_Avg_Csh_Lmt_Utl	当月取现额度使用率
R3m_Avg_Csh_Lmt_Utl	近 3 个月平均取现额度使用率
R6m_Avg_Csh_Lmt_Utl	近 6 个月平均取现额度使用率
R12m_Avg_Csh_Lmt_Utl	近 12 个月平均取现额度使用率
R1m_Avg_Cns_Bal_Rat	当月消费余额占总余额比

3. 划分训练集和测试集

同线性回归一样，以上数据作为建模的训练集，将时间窗整体向后移动一个月，2015 年 1 月作为表现期，2014 年 1 月至 2014 年 12 月作为观察期，准备另一份数据作为验证集，用于验证模型的稳定性，时间窗如图 9-8 所示。采用向后移动的好处是，一旦模型通过验证投入应用，由于采用了距当前最近的数据，所以时效性就高。当然，如果采用向前移动的方法，训练集的表现期应该为 2015 年 1 月，验证集的时间窗为 2015 年 2 月。



图 9-8 验证集数据的时间窗

相关书籍中也有将同一个时间窗内的建模数据随机划分为训练集和验证集的方法，但不能验证模型在过了一段时间后是否还有效，而且当数据量足够大时，这种方法检验的实际上是随机数。按照时间窗平移的方法，则模拟了模型的应用，验证结果更可靠。

表 9-7 是建模宽表的部分样例，其中：Csr\_ID 是客户编号；Evt\_Flag 是目标变量，1 表示响应（表现期内申请了信贷产品）；G 是分组标志，“T”组数据为训练集，对应的观察期末 Sta\_Dte 是 2015/1/31，“V”组数据是验证集，对应的观察期末 Sta\_Dte 是 2014/12/31。Logistic 回归的建模宽表格式与线性回归相同，区别只是目标变量，线性回归的目标变量是连续值，Logistic 回归的是离散值。

表 9-7 Logistic 建模宽表（部分样例）

Csr_ID	G	Sta_Dte	Evt_Flag	Age	Gender	Dob	Credit	R3M_Cns_Cnt	...
1001	T	2015/1/31	1	42	F	98	0	23	

(续)

Csr_ID	G	Sta_Dte	Evt_Flag	Age	Gender	Dob	Credit	R3M_Cns_Cnt	...
1002	T	2015/1/31	0	37	F	89	778	0	
1003	T	2015/1/31	0	44	F	32	0	23	
1004	T	2015/1/31	0	40	M	53	0	6	
1005	T	2015/1/31	1	41	M	23	14 474	6	
1006	T	2015/1/31	0	47	M	7	0	-	
...									
1001	V	2014/12/31	1	42	F	98	2800	2	
1002	V	2014/12/31	0	37	F	89	3889	-	
1003	V	2014/12/31	0	44	F	32	0	9	
1004	V	2014/12/31	0	40	M	53	2567	7	
1005	V	2014/12/31	1	41	M	23	0	9	
1006	V	2014/12/31	0	47	M	7	0	2	
...									

### 9.3.3 数据预处理

有了宽表数据之后，调用 SAS Proc 步、EM 的 LR 节点或者 R 的 Logistic 回归模型包就可以直接建模了，但效果不够好。原因是这样的数据对于建模太粗放，Logistic 回归是有监督的机器学习，要体现出“监督”。下面重点介绍两项能决定 Logistic 回归模型效果的数据预处理步骤，其他通用的预处理可参阅第 4 章或相关文献。

#### 1. 变量优化分组

先介绍连续变量。例如，图 9-9a 中随着年龄的增长，响应客户的占比并没有呈现出增加或减少的规律，或者年龄这个  $x$  与目标变量  $y$  没有单调相关关系，是一个无关因素。事实真是这样吗？图 9-9b 是经过离散化分组后的年龄，显然，30s' 的客户响应率明显更高，20s' 和 40s' 的客户响应率一样低。当然，这是为了说明问题刻意设计的数据分布规律，实际并非如此，但确实有很多本来与目标变量没有单调相关关系的连续变量在建模的时候被弃用，而实际上经过离散化处理后，马上显示出相关性。

再介绍离散变量（主要是类别变量）。在建模的时候，会先把类别变量转换为哑变量。还是用一个例子来说明，假设有一个变量 Province，在建模前每个省都会被设计成一个变量，如表 9-8 所示。这就意味着，当一个类别变量有几百个选项时，会生成几百个哑变量，如果有数百个类别变量……这对计算资源是很大的挑战。还有就是局部过拟合的问题，当一个类别变量的选项过多时，势必会在一些选项上样本量特别少，在生成哑变量后，一旦表现出显著的分类能力，这个哑变量将进入模型，以后会用来预测，而我们知道过少的样





SAS EM 提供了 PROC SPLIT 这个节点, 以实现变量的离散化。由于要针对每个变量分别进行处理, 当变量数超过数百个时变得难以接受。SAS 宏的迭代功能很好地解决了这个问题, 这也是 SAS 编码的魅力。因代码量较大, 此处不赘述, 感兴趣的读者可以联系笔者获取。

## 2. 计算分组变量的 WOE 值和 IV 值

变量经过重构和优化分组后, 对于每一个变量使用什么作为新的分组的水平值, 1、2、3 这样的组号还是 A、B、C 这样的类别吗? 首先要说明的是, 这两种做法都是可以的, 但有另一种更好的做法, 就是计算分组后的 WOE 值, 以 WOE 值作为新的变量值。一则可以直接使用线性回归过程中的共线性检验方法, 二则可以大幅提升计算效率。关于 WOE 值的计算, 参阅第 4 章。

## 3. 共线性检验

共线性检验的过程同线性回归一样, 可依据方差膨胀因子 Vif 从大到小逐个剔除变量, 直至 Vif 都小于 5, 参阅代码清单 4-7 及相关内容。

## 9.3.4 建模

### 1. 参数设置

经过数据预处理, 最终得到一组与目标变量有一定相关性、彼此独立且分布经过优化的候选变量 (一般会保留 50 ~ 100 个), 剩下的就是挑选显著变量拟合模型。这里使用 SAS 的 Proc Logistic 过程步实现, 如代码清单 9-1 所示。表 9-9 对最常用的几个参数作了重点介绍。

代码清单 9-1

```
Proc Logistic data= train_woe out=train_stat;  
Model Evt_Flg (Event='1') = &Var_Woe /  
selection=stepwise sle=0.05 sls=0.05;  
Score data= train_woe out= score_p;  
run;
```

表 9-9 Proc Logistic 的参数含义

参 数	含 义
data= train_woe	指定数据宽表, 变量是经过优化分组后的
out= train_stat	把建模的统计指标输出到指定的数据集 train_stat
model Evt_Flg (Event='1')=&Var_Woe	指定目标变量和预测变量, 预测 Evt_Flg=1 发生的概率, &Var_Woe 是经过层层把关的候选变量, 30 ~ 50 个为宜

(续)

参 数	含 义
selection=stepwise sle=0.05 sls=0.05	指定显著变量的检验方法 (参见第 8 章), 最常用的还是逐步筛选法 (STEPWISE), 使用 sle 和 sls 两个检验指标 <ul style="list-style-type: none"> <li>sle, 全称为 slentry, 规定变量进入模型的显著性水平, sle=0.05 表示 F 检验的 <math>P&lt;0.05</math> 时变量显著。</li> <li>sls, 全称为 slstay, 规定变量从模型中剔除的显著性水平, sls=0.05 表示 <math>P&lt;0.05</math> 时变量被剔除</li> </ul>
score data= train_woe out= score_p;	模型拟合之后对原数据集打分, 结果输出到 score_p 数据集 显然, 这里也可以指定测试数据集或者其他需要预测的数据集, 将建模、应用模型预测放在一个过程步实现

## 2. 结果解读

运行代码清单 9-1 这段代码后, 观察输出结果, 最主要的就是表 9-10 中的三张表。前两张表是经过筛选的显著变量以及参数估计值, 这里  $\text{logit}(\text{Evt\_Flg}) = -2.57 - 1.15 * \text{WOE\_CREDIT} + \dots$ , Wald 卡方显示每个变量的显著性水平, 值越大越显著。强调一点, 这个模型计算出来的  $y$  不是预测目标  $\text{Evt\_Flg}$ , 需要反  $\text{logit}$  转换得到概率。当然, 并不需要人工计算。第三个表显示出模型的区分能力, 如果目标变量实际是 1, 预测值也是 1, 或者实际是 0, 预测值也是 0, 那么预测与实际一致, 对应的就是表中“一致部分所占百分比”这个指标, 值越大, 表示模型区分 0 和 1 的能力越强。

表 9-10 Proc Logistic 运行后的 SAS 输出

最大似然估计值分析					
参数	自由度	估计值	标准误差	Wald 卡方	Pr > 卡方
Intercept	1	-2.5722	0.0862	890.1321	<0.0001
WOE_CREDIT	1	-1.1492	0.0826	193.4013	<0.0001
WOE_DEPOSIT	1	-1.1072	0.0680	264.9096	<0.0001
WOE_R6M_AVG_CDT	1	-0.4964	0.0634	61.2310	<0.0001
WOE_R6M_AVG_DEP	1	-0.4822	0.0547	77.8387	<0.0001
WOE_R6M_AVG_ILT	1	-0.9834	0.1499	43.0201	<0.0001
WOE_R6M_CLS_NBR	1	1.2512	0.2488	25.2844	<0.0001
WOE_R6M_CSH_AMT	1	-1.2574	0.1382	82.7421	<0.0001
WOE_R6M_TXN_AMT	1	-0.9406	0.0889	111.9310	<0.0001
WOE_R6M_TXN_CNT	1	-1.0364	0.1848	31.4389	<0.0001

优比估计值		
效应	点估计值	95% Wald 置信限
WOE_CREDIT	0.317	0.269 0.373
WOE_DEPOSIT	0.330	0.289 0.378
WOE_R6M_AVG_CDT	0.609	0.538 0.689
WOE_R6M_AVG_DEP	0.617	0.555 0.687
WOE_R6M_AVG_ILT	0.374	0.279 0.502
WOE_R6M_CLS_NBR	3.495	2.146 5.691
WOE_R6M_CSH_AMT	0.284	0.217 0.373
WOE_R6M_TXN_AMT	0.390	0.328 0.465
WOE_R6M_TXN_CNT	0.355	0.247 0.510

预测概率和观测响应的关联		
一致部分所占百分比	96.7 Somers D	0.938
不一致部分所占百分比	3.0 Gamma	0.941
结值百分比	0.3 Tau-a	0.112
对	5964231 c	0.969

表 9-11 是预测结果, 在表 9-7 建模宽表的基础上增加了三列。P\_1 是预测每个客户 Evt\_Flg=1 的概率, F\_Target 是 Evt\_Flg 的实际值, I\_Target 是根据 P\_1 的预测值, 表 9-10 中“一致部分所占百分比”指的就是这两个指标一致的比例。

表 9-11 Proc Logistic 模型的预测结果

Csr_ID	G	Sta_Dte	Evt_flag	F_Target	I_Target	P_1
1001	T	2015/1/31	1	1	1	0.819
1002	T	2015/1/31	0	0	0	0.383
1003	T	2015/1/31	0	0	0	0.259
1004	T	2015/1/31	0	0	0	0.457
1005	T	2015/1/31	1	1	0	0.443
1006	T	2015/1/31	0	0	0	0.106
...						
1001	V	2014/12/31	1	1	1	0.991
1002	V	2014/12/31	0	0	0	0.274
1003	V	2014/12/31	0	0	1	0.901
1004	V	2014/12/31	0	0	0	0.343
1005	V	2014/12/31	1	1	1	0.679
1006	V	2014/12/31	0	0	0	0.103
...						

### 3. 生成打分代码

模型建立之后, 面临的的就是如何固化下来应用。说白了, 模型就是公式  $\text{logit}(\text{Evt\_Flg}) = -2.57 - 1.15 * \text{WOE\_CREDIT} + \dots$ , 按公式里的变量准备好数据, 应用公式计算出结果再转换为概率打分, 就是预测。通常有以下三种方法。

方法 1: 输出参数估计值 (表: 最大似然估计值分析)。应用时, 预测哪些客户, 就建立一张与训练集一样的数据宽表, 只保留显著变量即可, 应用 SAS 的打分过程步来预测。

方法 2: 把训练集保存下来, 只保留模型显著变量。同样, 应用的时候按照这个样式准备好数据宽表, 只要显著的 x 变量即可。运行一遍上面的 Proc Logistic 建模过程即可, 设定 “selection=none”, 无需再做显著性检验。

方法 3: 上面两个方法的问题是, 需要对数据集重新做数据预处理, 且打分方式难以部署到应用系统。所以最佳方法是输出打分代码, 一组 if then 打分规则, 建立原始建模变量和目标变量间的关系。格式如代码清单 9-2 所示。

代码清单 9-2

```
lnp=0;
if .<CREDIT<0.00917510 then lnp=lnp+-0.0049253294;
```

```

else if 0.00917510<=CREDIT<0.02121887 then lnp=lnp+-0.4167213349;
else if 0.02121887<=CREDIT<0.03061749 then lnp=lnp+0.1062473556;
else if 0.03061749<=CREDIT<0.08684423 then lnp=lnp+-0.2715301891;
else if 0.08684423<=CREDIT<0.09815779 then lnp=lnp+-0.0825429578;
else if 0.09815779<=CREDIT<232.17477566 then lnp=lnp+-0.7977871337;
else if 232.17477566<=CREDIT<1166.75904702 then lnp=lnp+-0.2974839153;
else if 1166.75904702<=CREDIT<2022.25739051 then lnp=lnp+-0.6057602598;
else if 2022.25739051<=CREDIT<3088.50913985 then lnp=lnp+1.0345661202;
else if 3088.50913985<=CREDIT<4218.93071790 then lnp=lnp+2.0130888538;
else if 4218.93071790<=CREDIT<9330.91590793 then lnp=lnp+3.9405098135;
else if 9330.91590793<=CREDIT then lnp=lnp+2.0927377744;
if .<R6M_CLS_NBR<0.50000000 then lnp=lnp+0.1646394050;
else if 0.50000000<=R6M_CLS_NBR then lnp=lnp+-0.0114221470;
...
lnp=lnp-2.5722;
pl=1-exp(lnp)/(1+exp(lnp));

```

### 9.3.5 模型评估

模型建好了，效果如何？相比“一致部分所占百分比”这个指标，通常使用拟合曲线、ROC 曲线、KS、LIFT 四个评估指标。这些指标都来源于表 9-12 中的前四列数据。

模型的宗旨是总结客户的共性规律，把目标客户缩小到尽可能小的范围内，例如，原来针对 100 万个客户做营销有 10 000 个客户响应，用模型筛选出 20 000 个客户做营销就能有 8000 个响应，因此预测结果适用于客户群，评估模型也应站在客户群的角度。应用上面的打分代码对测试集（记住，不是训练集，这样的评估结果更客观）打分，预测每个客户的响应率 P\_1，按照从大到小排序，将全部客户等分为 20 组，然后计算每组客户的实际响应率（Actual，Evt\_Flg=1 占的比例）和预测响应率（Pred，P\_1 的均值），后面几列指标均从这四列计算而来。

表 9-12 模型评估指标的计算

Decile	样本量 N	实际响应率 Actual	预测响应率 Pred	响应数量 Response	未响应数量 NonResponse	累计响应率 CulResp%	累计未响应率 CulNonResp%	KS	不使用模型 Benchmark	LIFT
1	26 445	77%	78%	20 293	6152	16.0%	1.5%	14.5%	5%	3.21
2	26 444	72%	72%	18 948	7496	31.0%	3.4%	27.6%	10%	3.10
3	26 444	64%	64%	16 837	9607	44.3%	5.8%	38.5%	15%	2.95
4	26 444	42%	43%	11 204	15 240	53.1%	9.6%	43.6%	20%	2.66
5	26 444	25%	24%	6607	19 837	58.4%	14.5%	43.9%	25%	2.33
6	26 444	23%	23%	6049	20 395	63.1%	19.6%	43.6%	30%	2.10
7	26 445	22%	21%	5775	20 670	67.7%	24.7%	43.0%	35%	1.93
8	26 444	20%	20%	5359	21 085	71.9%	30.0%	42.0%	40%	1.80
9	26 444	18%	18%	4816	21 628	75.7%	35.3%	40.4%	45%	1.68



(续)

Decile	样本量 N	实际响应率 Actual	预测响应率 Pred	响应数量 Response	未响应数量 NonResponse	累计响应率 CulResp%	累计未响应率 CulNonResp%	KS	不使用模型 Benchmark	LIFT
10	26 444	17%	17%	4446	21998	79.2%	40.8%	38.4%	50%	1.58
11	26 444	16%	15%	4209	22235	82.6%	46.3%	36.2%	55%	1.50
12	26 444	15%	14%	3919	22525	85.7%	51.9%	33.7%	60%	1.43
13	26 444	14%	13%	3665	22779	88.6%	57.6%	31.0%	65%	1.36
14	26 445	12%	12%	3125	23 320	91.0%	63.4%	27.6%	70%	1.30
15	26 444	10%	11%	2736	23 708	93.2%	69.3%	23.9%	75%	1.24
16	26 444	9%	9%	2402	24 042	95.1%	75.3%	19.8%	80%	1.19
17	26 444	8%	8%	2112	24332	96.8%	81.3%	15.4%	85%	1.14
18	26 444	6%	7%	1670	24774	98.1%	87.5%	10.6%	90%	1.09
19	26 444	5%	6%	1426	25018	99.2%	93.7%	5.5%	95%	1.04
20	26 444	4%	4%	1018	25426	100.0%	100.0%	0.0%	100%	1.00
合计	528 883			126 616	402 267					

### 1. 拟合曲线

图形能够最直观地展示模型的拟合效果(见图9-10),能够展示总体预测误差。拟合曲线就是预测响应率 vs 实际响应率,即表9-11中的第三、四两列。有以下三个解读视角。

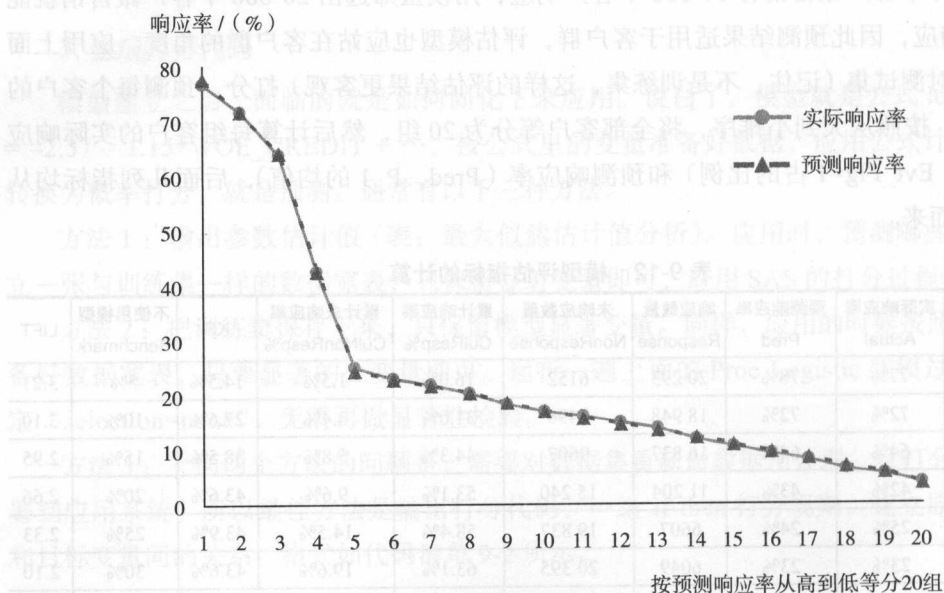


图9-10 预测模型的拟合曲线

1) 两条曲线尽可能地接近,说明总体预测偏差小,预测准确。

2) 曲线下降的幅度大, 说明模型区分能力强, 模型变量抓住了响应客户的主要特征, 能够将目标客户识别出来。

3) 曲线走势平滑, 说明预测响应率高的客户的确响应率更高, 无过拟合问题, 模型较为稳定。

## 2. ROC 曲线

ROC, 即累计边际响应率, 相比不使用模型, 应用模型能够缩小目标客户分布的范围。在没有预测打分的情况下, 将全部客户等分 20 组, 每组将涵盖 5% 的响应客户, 如果有预测打分, 就可以将潜在响应率高的客户集中到少数几组内。图 9-11 最可能响应的两组客户 (占总量的 10%) 覆盖了 83% 的实际响应客户。换言之, 可针对这 10% 的客户开展营销, 其效果和针对全部客户的差不多。营销成本大幅降低的同时更快地响应客户需求, 或者将营销成本集中分配给更少的客户, 效果必然更好。

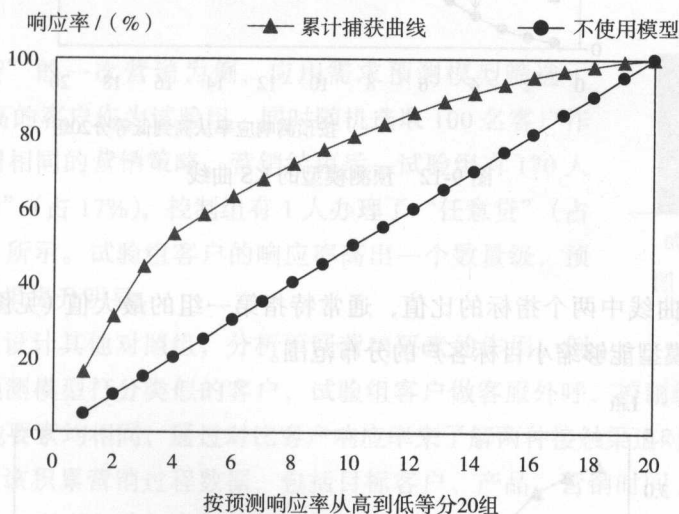


图 9-11 预测模型的 ROC 曲线

## 3. KS

$KS = \text{累计响应率} - \text{累计未响应率}$ , 可以量化评估模型的区分度, 通常使用 20 组中最大的那个值。KS 的经验值为:

20% 以下, 区分度弱;

20% ~ 30%, 区分度较弱;

30% ~ 40%, 区分度较强;

40% ~ 60%, 区分度非常强;

60%以上, 区分度过强, 需要检查变量的设计是否存在问题, 例如因果倒置。

如图 9-12 所示, KS 为图中两条曲线的 Gap, 最大值出现得早, 则能更好地缩小目标范围。KS 曲线和 ROC 曲线的差异在于对比曲线, 前者对比累计未响应曲线, 后者对比不使用模型的随机响应曲线。

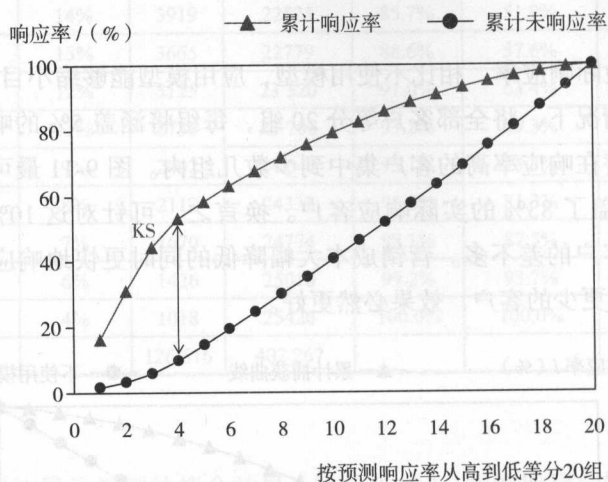


图 9-12 预测模型的 KS 曲线

#### 4. Lift

Lift 是 ROC 曲线中两个指标的比值, 通常特指第一组的最大值 (见图 9-13)。相比不使用模型, 应用模型能够缩小目标客户的分布范围。

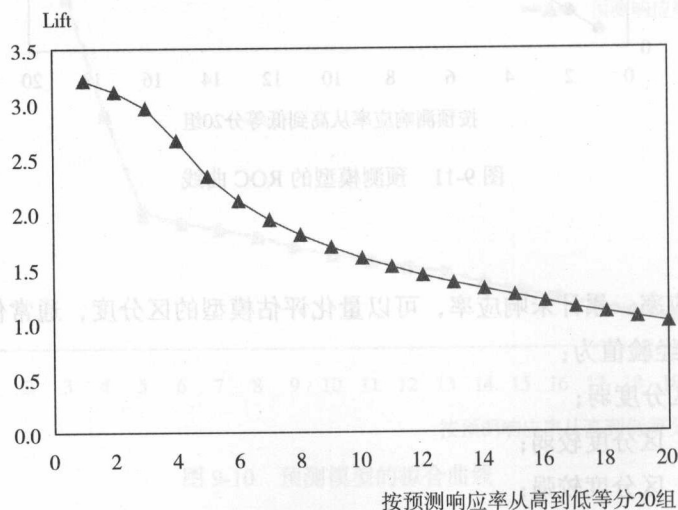


图 9-13 预测模型的 Lift 曲线

综上所述, 这四个指标相辅相成, 全面评估模型的稳定性、准确性和区分度, 为当前大部分公司所使用。

## 9.4 预测模型的应用与评估

模型构建之后, 部署应用。定期(根据时间窗的粒度, 这里为每个月初)调用模型, 预测每个客户接下来一个月内的潜在响应率, 每天筛选最可能申请的少量客户来营销, 以提高单次营销成功率。

为了不断优化营销效果, 会在营销过程中做对比试验, Test-Control 分析是常用的方法, 即设计试验组和控制组, 使其中一个营销要素不同而其他要素均相同, 通过对比两组客户的营销效果, 分析决定营销效果的关键因素, 优化策略。有时甚至会在一次营销过程中设计上百个对照组, 以便分析目标客户、接触渠道、接触时间、话术、offer 等各项营销要素的提升空间。

以“任意贷”的一次营销为例, 应用需求预测模型筛选了 1000 名打分最高的客户作为试验组, 同时随机选取 100 名客户作为控制组, 采用相同的营销策略。营销结束后, 试验组有 170 人办理了“任意贷”(占 17%), 控制组有 1 人办理了“任意贷”(占 1%), 如图 9-14 所示。试验组客户的响应率高出一个数量级, 预测模型对营销效果提升明显。

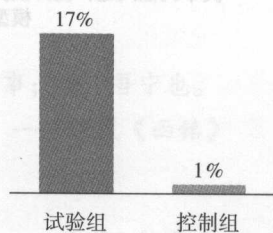


图 9-14 Test-Control  
营销效果评估

同样, 可以设计其他对照组, 分析不同营销要素的作用。例如, 筛选需求预测模型打分类似的客户, 试验组客户做客服外呼, 控制组客户利用短信推送, offer 等其他要素均相同, 通过对比客户响应率来了解两种接触渠道对营销效果的影响。

同时, 还应该积累营销过程数据, 包括目标客户、产品、营销时间、联系渠道、营销结果、客服人员、客户申请与审批结果等, 用于评估模型和各项营销设置的改进空间, 使得整个营销活动得到持续优化。

## 9.5 本章小结

线性回归和 Logistic 回归是最常用的两种预测方法, 分别用于预测连续数值和事件发生的概率。算法本身和模型的训练过程并不复杂, 复杂的是前期的数据预处理, 从一定程度上来说, 这决定了模型拟合的质量。当然, 数据预处理是可选项, 如果对模型效果没有特别的追求, 建模仅需三步: 准备数据、模型训练(即显著性检验)和效果评估。

在模型训练过程中, 会有大量的统计学指标从各个角度评估变量及模型的效果, 一定



要记住，这些指标仅反应模型对训练集本身的拟合情况，模型真正的效果体现在能否准确、可靠、稳定地应用于预测，所以直接对比预测结果和实际值是最好的评估方法。

总体来说，在应用该算法时应关注 7 点，如图 9-15 所示。



图 9-15 Logistic 回归算法关键提升点

## 决策树类算法，反欺诈模型“专家”

富贵福泽，将厚吾之生也；贫贱忧戚，庸玉汝于成也。存，吾顺事；没，吾宁也。

——张载《西铭》

你不能衡量它，就不能管理它。

——彼得·德鲁克

很多预测问题的本质是分类，例如营销预测模型的目标变量多是客户“买”还是“没买”，风险预测模型的目标变量则是客户“拖欠”或“未拖欠”，客户偏好的渠道是“网银”“手机”还是“微信”，诸如此类的问题皆为“分类”。Logistics 回归是解决“分类”问题的专家，尤其擅长分析线性关系，对数据整体结构的把握良好，能够给出每条记录的评价。决策树类算法（包括各类决策树、随机森林、深度学习等）则是更为重要的分类模型和机器学习专家，采用分割的方法，层层递进，能够深入局部数据，擅长处理非线性关系。目前，决策树类算法被广泛应用于信用评分和欺诈识别。本章结合客户画像和电商的反盗卡欺诈案例，介绍最为简单的决策树模型，并附带介绍决策树的集群应用——随机森林，基于书中提供的决策树模型构建代码，读者通过增加随机抽样可以很容易实现随机森林算法。

### 10.1 决策树，重要的分类器

分类是数据挖掘领域最为重要的一类问题，分类器则是分类方法的统称，包括决策树、Logistic 回归、朴素贝叶斯、神经网络等算法。同时，大部分预测问题本质上也是分类，通

过对比“好”客户和“坏”客户，归纳各自的特征，对其他具有类似特征的客户，计算“好”或“坏”的程度，即为预测，过程如图 10-1 所示。

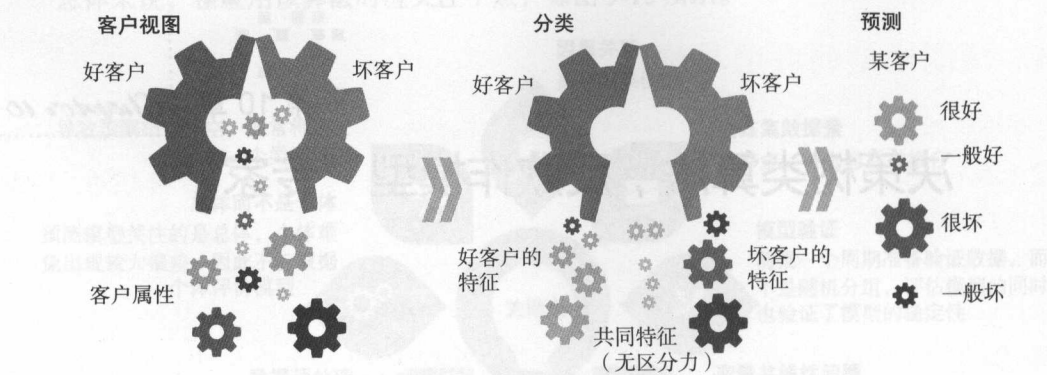


图 10-1 分类过程示意图

决策树，简单来说，是一种树形结构的分类器，采用自上而下的递归方式，对全部数据组成的空间进行分割，划分为有限的组群。决策树方法广泛应用于客户群细分、欺诈识别与信用评分等领域。

构建一棵决策树，关键问题就在于，选择哪些属性进行分割、如何分割以及怎样处理不合适的分支。常见的决策树算法有 ID3、C4.5、C5.0、CART 和 CHAID，其主要差别也正在于处理这些关键问题的方法，如表 10-1 所示。

表 10-1 常见决策树算法对比

算法	分割规则	分支数量	修剪规则
ID3	信息增益	多叉树	无剪枝
C4.5/C5.0	信息增益率	多叉树	后剪枝
CART	Gini 指标、熵	二叉树	后剪枝
CHAID	卡方检验	多叉树	前剪枝

其中，分类和回归树（CART）是众多决策树算法中使用最为广泛的一种，10.2 节结合客户画像介绍 CART 是如何实现分类的，其他算法在分析思路上大同小异。

## 10.2 决策树的关键思想

本节以客户画像为例，介绍决策树算法的两大关键思想。

### 10.2.1 理财客户画像案例背景

先介绍客户视图和客户画像。客户视图是按照客户 ID 汇总每个人全部的信息和数据，

包括年龄、性别等基本信息, 交易金额、来电次数等统计指标, 以及在这些信息基础上交叉衍生出的变量 (参见第4章), 表现形式上以宽表为主。客户画像则是围绕某个目标, 在客户视图提供的信息中归纳出目标客户所具有的显著特征, 例如某类理财产品的客户画像是35~45岁的代发工资客户, 信用卡流失的客户画像是半年内消费明显下降且来电增多的客户。客户画像既是特征分析的过程, 也是结果。

银行想对某类理财产品的个人客户进行画像, 发现购买理财产品的客户具有什么样的特征。表10-2是个人客户视图的样例数据, 包括三列: 近三个月信用卡消费额 (Sal\_Mth)、近三个月借记卡日均活期存款余额 (Flw\_Bal\_Dal)、截至当前有没有购买该类理财产品 (Buy=1/Non=0)。

表 10-2 某银行理财产品的客户数据样例

Csr_ID	Sal_Mth	Flw_Bal_Dal	Buy=1/Non=0
1	7000	28 400	1
2	9550	26 800	1
3	7480	31 600	1
4	7150	30 800	1
5	9700	33 600	1
6	12 010	29 200	1
7	11 800	27 600	1
8	9280	32 400	1
9	7900	30 000	1
10	10 300	30 800	1
11	6100	32 000	1
12	9100	30 000	1
13	8500	29 600	0
14	6280	30 800	0
15	7480	27 200	0
16	5320	30 400	0
17	9400	27 600	0
18	5920	27 600	0
19	6940	26 000	0
20	7600	28 400	0
21	5740	26 400	0
22	4300	28 800	0
23	6100	24 000	0
24	7300	24 800	0



## 10.2.2 关键思想一：递归划分

在这个分类问题中，自变量是近三个月信用卡消费额 ( $x_1$ )、近三个月借记卡日均活期存款余额 ( $x_2$ )，要分类的目标变量是截至当前有没有购买该类理财产品 ( $y$ )， $y$  有两个选项 (购买和未购买)。

递归划分，是通过递归的方式把关于变量  $x$  的  $p$  维空间 (这个例子中  $p=2$ ) 划分为不重叠的矩形。首先，使用一个变量  $x$  把空间划分为 2 个矩形，让每个矩形里的  $y$  尽可能是一类，接着用另一个变量  $x$  对某一个小矩形继续划分，直到每个小矩形里的点尽可能是一类 (提纯)。

首先绘制散点图，观察数据分布。将表 10-2 的数据绘制成散点图，可以清晰地看到由  $x_1$ 、 $x_2$  构成的二维空间，用不同颜色的点标识出购买者和未购买者，如图 10-2 所示。这也是为什么本例只选两个  $x$  变量，因为容易展示。接下来看看算法是如何分割的。

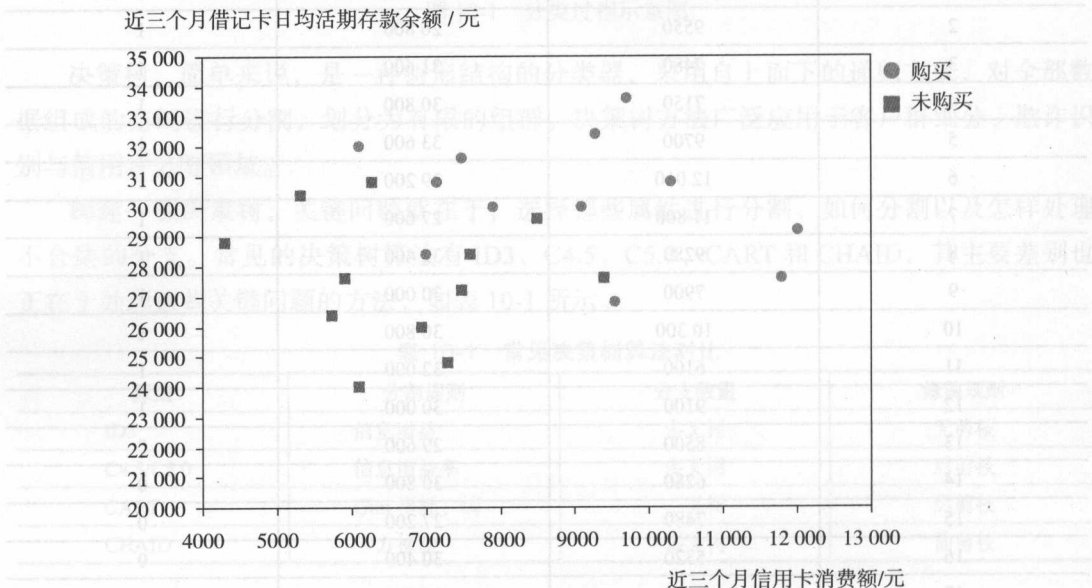


图 10-2 购买 vs 未购买客户分布图

### 1. 算法的第 1 次分割

算法首先选择  $x_2=29\ 000$  元做第 1 次分割。由  $x_1$ 、 $x_2$  组成的空间被划分为两个矩形 (见图 10-3)，每个矩形比分裂前更同质：上面的矩形绝大多数是购买者 (9 vs 3)，下面的矩形绝大多数是未购买者 (9 vs 3)。

### 2. 算法的第 2 次分割

算法其次选择  $x_1=9475$  元做第 2 次分割。如图 10-4 所示，左下角的矩形包含满足

$x_1 \leq 9475$  元 &  $x_2 \leq 29\ 000$  元的点, 除了一个是购买者之外, 其余都是未购买者; 右下角的矩形包含满足  $x_1 > 9475$  元 &  $x_2 \leq 29\ 000$  元的点, 包含两个购买者。

近三个月借记卡日均活期存款余额 / 元

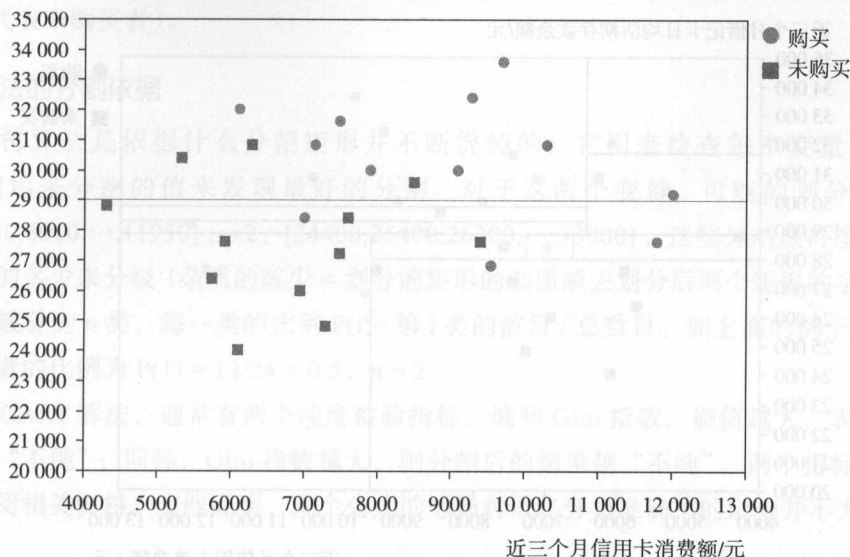


图 10-3 算法的第 1 次分割

近三个月借记卡日均活期存款余额 / 元

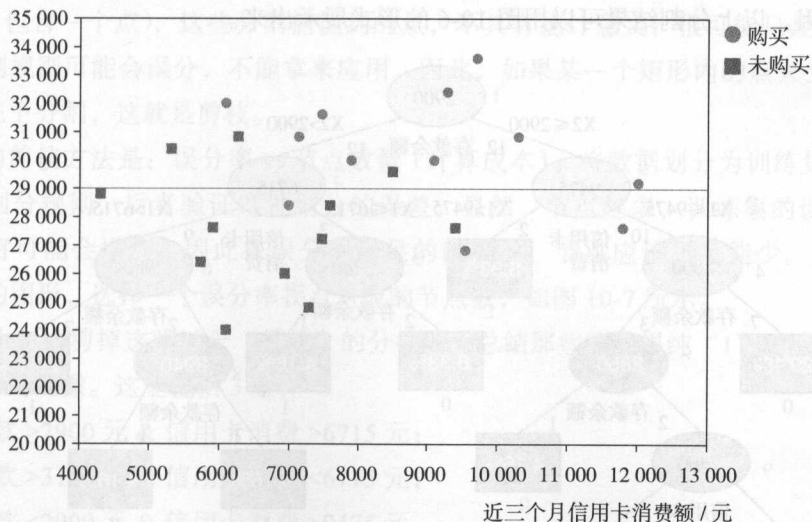


图 10-4 算法的第 2 次分割

### 3. 接下来的分割

接下来算法做第3次、第4次、……、第n次分割，不断划分矩形，使之变得更纯，最终每个矩形中的点是相同的（圆形、正方形完全区分开），如图10-5所示。

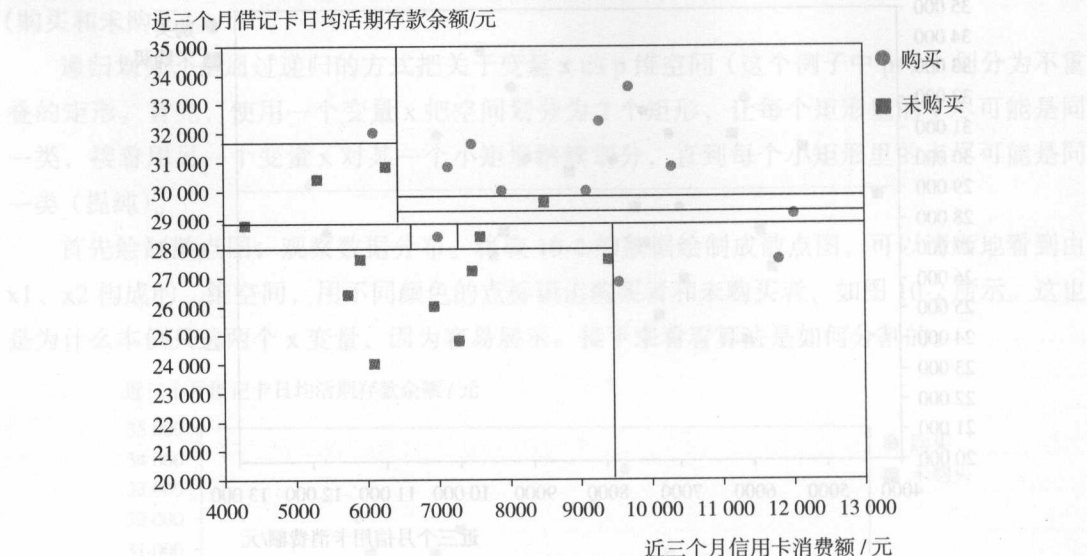


图 10-5 算法的最终分割结果

从以上过程可见，每次分割都可以描述为把一个节点分成两个后续节点，这也是称为分类树的原因。以上分割结果可以用图10-6的形式展示出来。

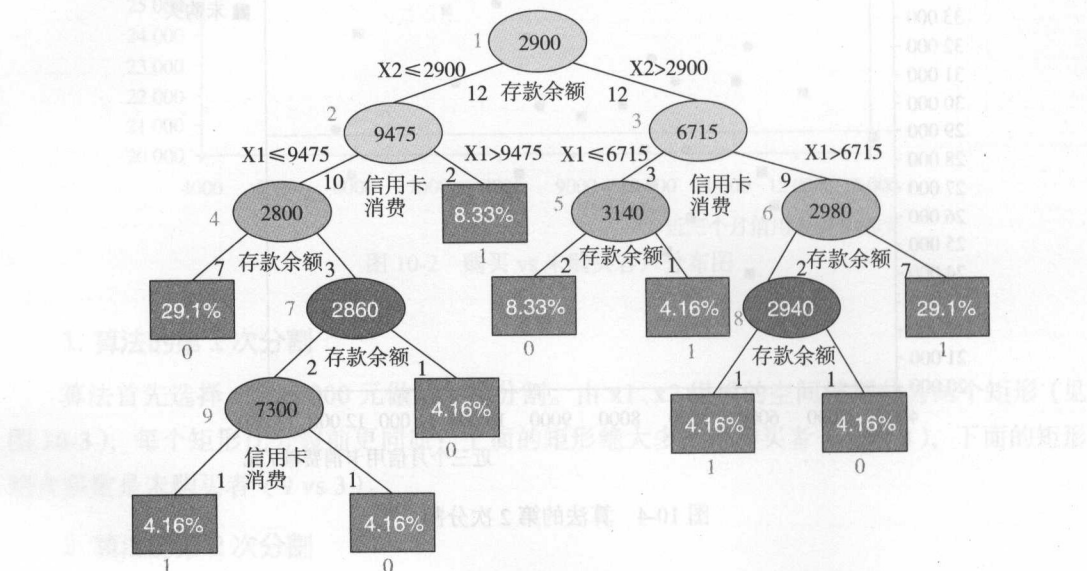


图 10-6 客户画像分类树（单位：元）

图 10-6 中: 圆圈下的文字为划分依据  $x$ ; 圆圈里的数字为变量  $x$  的分割点; 圆圈左侧的数字为分割的先后顺序; 线下数字为点 (客户) 的数量; 矩形为最终被提纯的节点; 矩形里的数字为矩形内的点占总体的比例, 合计为 100%; 矩形下的数字为  $y$  的类别 (1 代表购买者, 0 代表未购买者)。

#### 4. 算法的分割依据

决策树算法是依据什么分割矩形并不断提纯的? 它用来检查每个变量和该变量所有可能用来分割的值来发现最好的分割。对于这两个变量, 可能的划分点是  $x_1: \{4810, 5530, 6010, \dots, 11950\}$ ,  $x_2: \{24400, 25400, 26200, \dots, 33000\}$ , 这些分割点再按照它们能减少杂质的多少来分级 (杂质的减少 = 划分前矩形的杂质减去划分后两个矩形的杂质之和)。如果记录被分为  $n$  类, 每一类的比例  $P(i) = \text{第 } i \text{ 类的数目} / \text{总数目}$ 。如上面的例子, 24 个样本中购买者的比例为  $P(1) = 12/24 = 0.5$ ,  $n = 2$ 。

对于 CART 算法, 通常有两个纯度检验指标, 熵和 Gini 指数。熵值越大, 表示分割后的结果越“不纯”; 同样, Gini 指数越大, 则分割后的结果越“不纯”。两个指标的计算公式可以查阅相关资料, 实践证明, 两个公式的选择对最终分类准确率的影响并不大。

### 10.2.3 关键思想二: 剪枝

为什么要剪枝? 在上面的例子中, 最后几个分割导致矩形中有很少的点 (事实上, 有 6 个矩形只包含一个点), 这些分割捕捉到的点, 不具有统计意义, 很可能只是异常点, 因此这种分割规则可能会误分, 不能拿来应用。因此, 如果某一个矩形内的点太少, 就删掉, 或者不再往下分割, 这就是剪枝。

常用的剪枝方法是: 误分率 vs 节点数量 (计算成本)。将数据划分为训练集和验证集, 前者生成划分规则, 后者验证这种划分的误差。显然, 节点越多, 训练集的误分率越低, 但验证集有可能会增加, 因此在误分率满足的前提下, 节点应该尽可能少。具体可根据 SAS 输出的图形, 选择一个误分率拐点对应的节点数, 如图 10-7 所示。

针对图 10-6 剪掉这些过于“柔软”的分枝后, 总结那些能够提纯“1”的分枝特征, 就是理财客户的画像。这里包括 3 条:

- 1) 存款 > 2900 元 & 信用卡消费 > 6715 元;
- 2) 存款 > 3140 元 & 信用卡消费 < 6715 元;
- 3) 存款 < 2900 元 & 信用卡消费 > 9475 元。



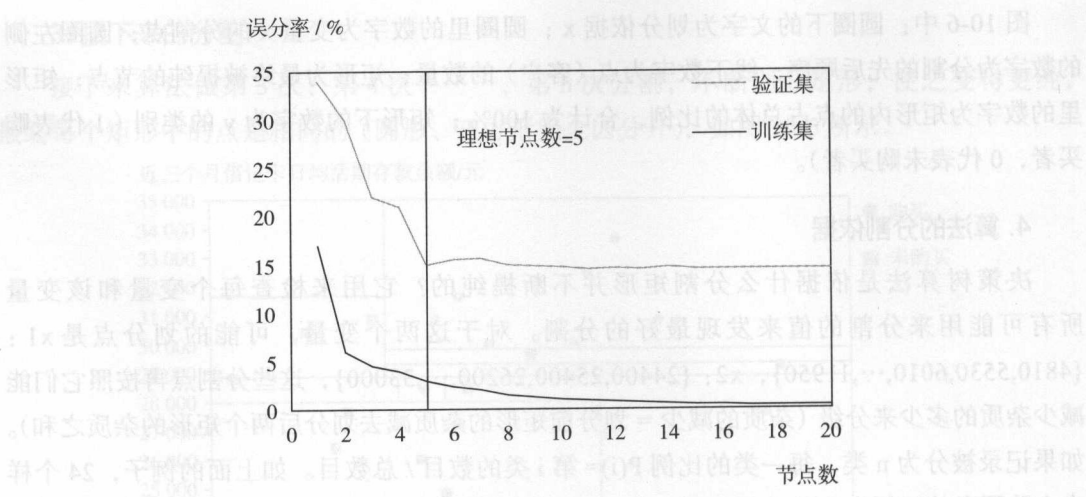


图 10-7 剪枝方法：误分率 vs 节点数

## 10.3 案例：电商盗卡交易风险识别

### 10.3.1 案例背景

随着互联网科技的日新月异，电子商务的蓬勃发展，各种网上交易平台如雨后春笋般涌现出来，随之而来的是交易中的各种欺诈风险，以跨境支付为例，最大的风险就是盗卡交易风险。

所谓盗卡交易风险，是指不法分子通过非法手段获取了信用卡信息，经由网上交易的方式非法获利。这极大地损害了平台的利益，同时也给信用卡持卡人带来困扰。因此，需要一套风控系统来识别并防控此类风险，整个风控系统非常复杂，其中会有很多模块。以单个模块来看，其最主要的构成是风控规则。一条风控规则，简单理解就是刻画某一种风险的逻辑，比如交易金额大于 10 000 元就拒绝当笔交易，可以用来覆盖单笔大额的风险。一般而言，风控规则的逻辑比较复杂，同时会涉及好几个变量。

如何设定风控规则呢？一是根据业务理解；二是通过数据分析。

某网上平台的一个支付场景下，近期盗卡交易占比有提升，现在准备补充一批规则，进行针对性的补防，共有交易 8000 余笔，相关变量如表 10-3 所示。

表 10-3 盗卡交易分析的部分变量

变量名	说 明
black_flg	欺诈标签
chg_card_cnt	换卡次数
chg_bank_country_cnt	换发卡国

(续)

变量名	说 明
chg_email_cnt	换联系邮箱数量
chg_seller_cnt	换卖家数量
ip_card_mismatch	IP 国和发卡国冲突
bank_fail_cnt	银行失败次数
pay_succ_cnt	支付成功次数
chg_device_cnt	换设备次数
chg_ip_cnt	换 IP 次数
chg_ip_country_cnt	换 IP 国次数
amt	交易金额

下面结合案例，分别使用 SAS、Clementine、R 等软件来实现 CART、CHAD 和 C.50 算法。

### 10.3.2 以 SAS 实现

SAS 广泛使用于电商的反欺诈领域。除了使用 SAS EM 进行界面化的决策树分析外，使用 SAS 代码生成决策树的过程更易于应用。代码清单 10-1 是使用 SAS Split 过程步实现决策树构建的代码。

代码清单 10-1

```
Proc split data=step01 outleaf=leaf
outimportance=importance outtree=tree outmatrix=matrix outseq=seq
criterion=entropy
assess=impurity
maxbranch=10
maxdepth=10
exhaustive=5000
leafsize=10
splitsize=10
subtree=assessment;
code file="C:\Users\sas_rule.txt";
describe file="C:\rulefinal.txt";
input chg_card_cnt chg_bank_country_cnt chg_email_cnt chg_seller_cnt
ip_card_mismatch bank_fail_cnt pay_succ_cnt chg_device_cnt
chg_ip_cnt chg_ip_country_cnt amt / level=interval;
target black_flg/level=binary;
score data=step01 nodmdb out=score outfit=fit;
run;
```

其中，criterion 为分裂方法，可选 CHISQ（卡方检定）、entropy（信息增益率）、ERATIO（信息增益）、GINI（GINI 指标）等；maxdepth 为树的最大深度；maxbranch 为对于一个节点而

言最大的子节点数。模型运行完毕后,会生成规则的代码,代码清单 10-2 展示了其中一部分。

代码清单 10-2

IF 17.5 <= chg_card_cnt < 19.5			
THEN			
NODE	:	8	
N	:	19	
0	:	5.3%	
1	:	94.7%	
IF amt < 30.79			
AND chg_card_cnt < 1.5			
THEN			
NODE	:	12	
N	:	205	
0	:	100.0%	
1	:	0.0%	
IF 1227.41 <= amt < 1508.105			
AND chg_card_cnt < 1.5			
THEN			
NODE	:	20	
N	:	61	
0	:	100.0%	
1	:	0.0%	
IF 8.5 <= pay_succ_cnt			
AND 1.5 <= chg_card_cnt < 4.5			
THEN			
NODE	:	31	
N	:	160	
0	:	100.0%	
1	:	0.0%	
IF 4.5 <= bank_fail_cnt < 5.5			
AND 4.5 <= chg_card_cnt < 6.5			
THEN			
NODE	:	33	
N	:	19	
0	:	42.1%	
1	:	57.9%	
IF 8.5 <= bank_fail_cnt < 11.5			
AND 4.5 <= chg_card_cnt < 6.5			
THEN			
NODE	:	35	
N	:	20	
0	:	90.0%	
	:	10.0%	

### 10.3.3 以 Clementine 实现

使用 Clementine 软件建模不需要编写代码，而是通过拖曳的方式实现，门槛较低，可视化强。下面介绍决策树建模的几个重要步骤，更多操作细节可以参考相关书籍。

第一步，从底部的 Sources 中选取数据输入的格式，以及从 Modeling 中选择相应的算法，Clementine 可以提供 CART、CHAID 和 C5.0 等算法（见图 10-8），这里选择 CHAID 作为示范。

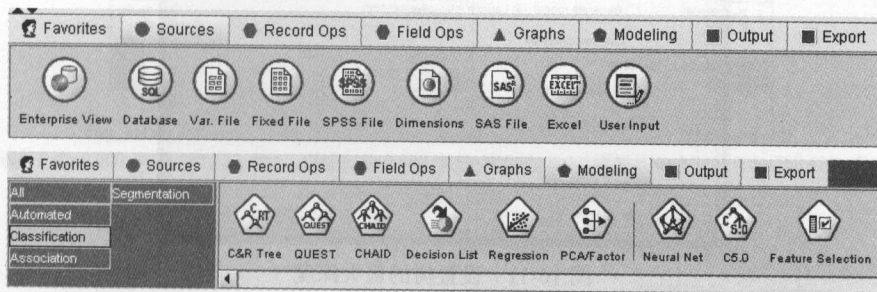


图 10-8 Clementine 可提供 CART、CHAID 和 C5.0 等算法

第二步，使用箭头连接数据源和算法，并进行相关设置（见图 10-9）。

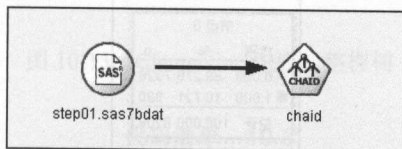


图 10-9 连接数据源和算法

在 CHAID 算法设置页面的 Fields 中，需要设定自变量和因变量（见图 10-10）。

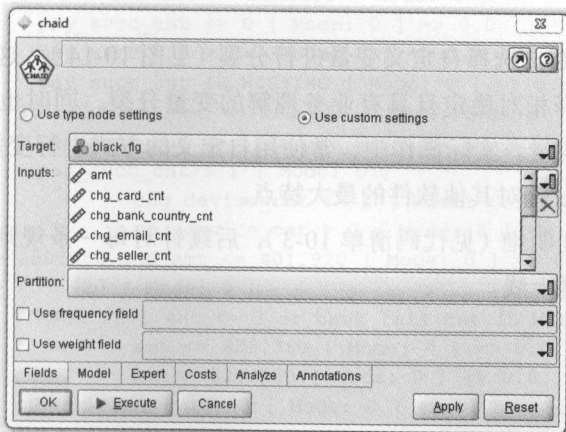


图 10-10 在 Fields 中设定自变量和因变量



在 Model 页面，选择交互功能，可以在建模过程中互动。另外，还可以设置树的最大深度，以进行前剪枝，本文设置为 10，如图 10-11 所示。

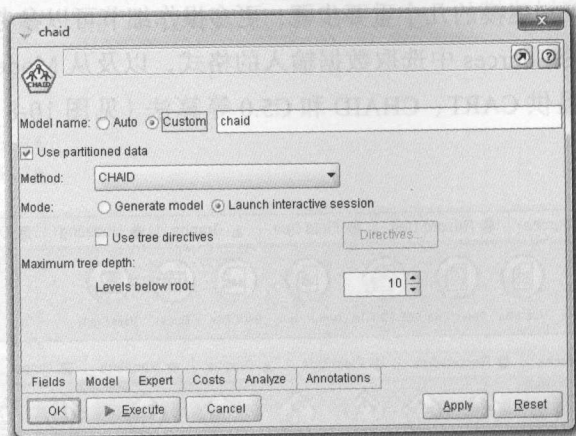


图 10-11 设置树的最大深度

第三步，运行模型，并实现人机互动。运行后，得到根节点（见图 10-12）。

black_flg			
节点 0			
类别	%	n	
0.000	89.279	7328	
1.000	10.721	880	
总计	100.000	8208	

图 10-12 得到根节点

此时有两个方式，一是右击选择生长树，这样系统会自动生成整棵树，如图 10-13 所示。

另一个方式是，右击选择自定义变量进行分裂（见图 10-14）。这样的好处是，可以根据业务理解，优先选择相对稳定且具有业务理解的变量分裂，同时也可以去除某些效能较好但是业务理解差的变量。实际操作中，常使用自定义的方式进行变量的人工筛选和剪枝。互动性也是 Clementine 针对其他软件的最大特点。

第四步，导出规则明细（见代码清单 10-3），后续针对每一条规则进行跨时间验证，选择效果较好的规则部署上线。

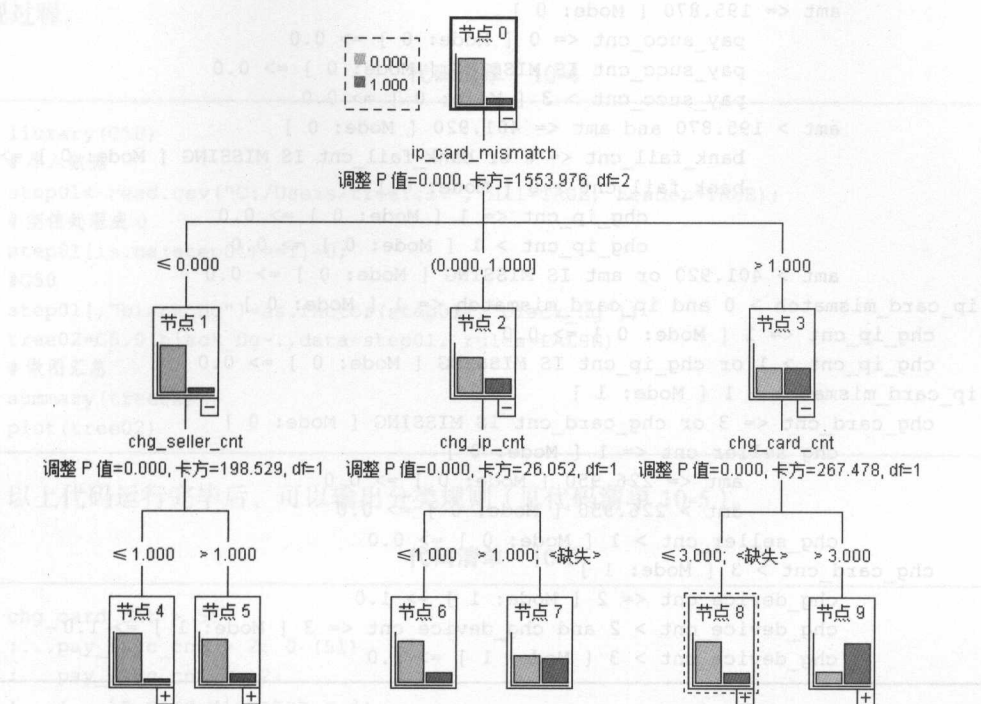


图 10-13 Clementine 生成的整棵树

## 代码清单 10-3

```

ip_card_mismatch <= 0 [ Mode: 0 ]
  chg_seller_cnt <= 1 [ Mode: 0 ]
    amt <= 55.090 or amt IS MISSING [ Mode: 0 ] => 0.0
    amt > 55.090 and amt <= 151.090 [ Mode: 0 ]
      pay_succ_cnt <= 0 [ Mode: 0 ] => 0.0
      pay_succ_cnt > 0 [ Mode: 0 ] => 0.0
      pay_succ_cnt IS MISSING [ Mode: 0 ] => 0.0
    amt > 151.090 and amt <= 290.560 [ Mode: 0 ]
      pay_succ_cnt <= 1 or pay_succ_cnt IS MISSING [ Mode: 0 ] => 0.0
      pay_succ_cnt > 1 [ Mode: 0 ]
        chg_device_cnt <= 2 [ Mode: 0 ] => 0.0
        chg_device_cnt > 2 [ Mode: 0 ] => 0.0
      amt > 290.560 and amt <= 401.920 [ Mode: 0 ] => 0.0
      amt > 401.920 [ Mode: 0 ]
        bank_fail_cnt <= 3 or bank_fail_cnt IS MISSING [ Mode: 0 ]
          amt <= 629.360 [ Mode: 0 ] => 0.0
          amt > 629.360 [ Mode: 0 ] => 0.0
        bank_fail_cnt > 3 [ Mode: 0 ] => 0.0
    chg_seller_cnt > 1 [ Mode: 0 ]

```

```

amt <= 195.870 [ Mode: 0 ]
    pay_succ_cnt <= 0 [ Mode: 0 ] => 0.0
    pay_succ_cnt IS MISSING [ Mode: 0 ] => 0.0
    pay_succ_cnt > 3 [ Mode: 0 ] => 0.0
amt > 195.870 and amt <= 401.920 [ Mode: 0 ]
    bank_fail_cnt <= 0 or bank_fail_cnt IS MISSING [ Mode: 0 ] => 0.0
    bank_fail_cnt > 0 [ Mode: 0 ]
        chg_ip_cnt <= 1 [ Mode: 0 ] => 0.0
        chg_ip_cnt > 1 [ Mode: 0 ] => 0.0
amt > 401.920 or amt IS MISSING [ Mode: 0 ] => 0.0
ip_card_mismatch > 0 and ip_card_mismatch <= 1 [ Mode: 0 ]
    chg_ip_cnt <= 1 [ Mode: 0 ] => 0.0
    chg_ip_cnt > 1 or chg_ip_cnt IS MISSING [ Mode: 0 ] => 0.0
ip_card_mismatch > 1 [ Mode: 1 ]
    chg_card_cnt <= 3 or chg_card_cnt IS MISSING [ Mode: 0 ]
        chg_seller_cnt <= 1 [ Mode: 0 ]
            amt <= 226.950 [ Mode: 0 ] => 0.0
            amt > 226.950 [ Mode: 0 ] => 0.0
            chg_seller_cnt > 1 [ Mode: 0 ] => 0.0
        chg_card_cnt > 3 [ Mode: 1 ]
            chg_device_cnt <= 2 [ Mode: 1 ] => 1.0
            chg_device_cnt > 2 and chg_device_cnt <= 3 [ Mode: 1 ] => 1.0
            chg_device_cnt > 3 [ Mode: 1 ] => 1.0

```

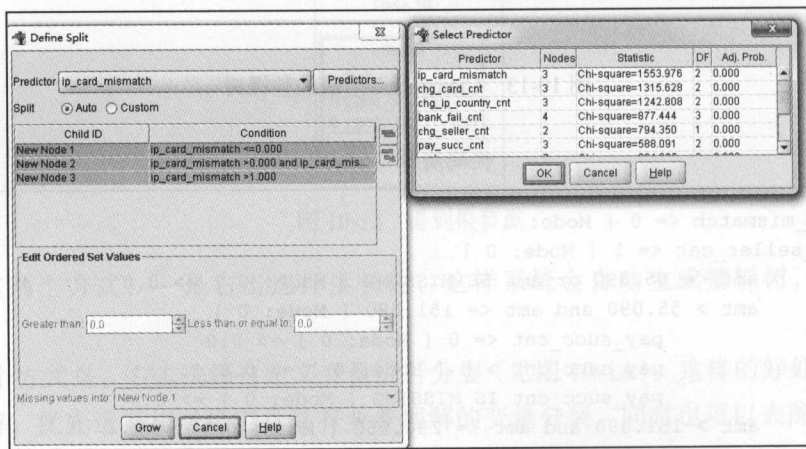


图 10-14 右击选择自定义变量进行分裂

以上代码简单演示了如何使用 Clementine 来实现基于 CHAID 算法的决策树，建模过程中 Clementine 提供了很多的选项，感兴趣的读者可以自行操作。

### 10.3.4 以 R 实现

R 是一种免费的开源统计分析软件，目前广受欢迎。代码清单 10-4 给出了 C5.0 算法的

实现过程。

代码清单 10-4

```
library(C50)
# 导入数据
step01<-read.csv("C:/Users/tree.csv", fill=TRUE, header=TRUE);
# 空值处理成 0
step01[is.na(step01)==T]=0;
#C50
step01[, "black_flg"]=as.factor(step01[, "black_flg"])
tree02=C5.0(black_flg~., data=step01, rules=FALSE)
# 做图汇总
summary(tree02)
plot(tree02)
```

以上代码运行完毕后, 可以输出分类规则 (见代码清单 10-5)。

代码清单 10-5

```
chg_card_cnt > 5:
...pay_succ_cnt > 2: 0 (51)
:   pay_succ_cnt <= 2:
:     ...ip_card_mismatch > 4:
:       ...chg_ip_cnt <= 10: 1 (287/24)
:       :   chg_ip_cnt > 10:
:       :     ...bank_fail_cnt <= 33: 0 (10)
:       :     :   bank_fail_cnt > 33: 1 (8/1)
:       ip_card_mismatch <= 4:
:       ...chg_seller_cnt > 1: 1 (63/14)
:       :   chg_seller_cnt <= 1:
:       :     ...chg_card_cnt > 15: 1 (3)
:       :     :   chg_card_cnt <= 15:
:       :     :     ...pay_succ_cnt <= 1: 0 (32/1)
:       :     :     :   pay_succ_cnt > 1: 1 (5/1)
chg_card_cnt <= 5:
...ip_card_mismatch <= 0:
...chg_seller_cnt <= 1: 0 (6080/256)
:   chg_seller_cnt > 1:
:     ...pay_succ_cnt > 5: 0 (140)
:     :   pay_succ_cnt <= 5:
:     :     ...amt > 188.58:
:     :       ...chg_email_cnt <= 5: 0 (485/100)
:     :       :   chg_email_cnt > 5:
:     :       :     ...chg_email_cnt <= 6: 0 (7/1)
:     :       :     :   chg_email_cnt > 6: 1 (12/2)
:     :       amt <= 188.58:
:     :       ...chg_seller_cnt > 2:
:     :       :   ...amt > 113.38: 0 (14)
:     :       :   :   amt <= 113.38:
```



```

:         : ....bank_fail_cnt <= 2: 1 (6)
:         :         bank_fail_cnt > 2: 0 (4/1)
:         chg_seller_cnt <= 2:
:         : ....chg_card_cnt <= 3: 0 (289/18)
:         :         chg_card_cnt > 3:
:         :         : ....chg_bank_country_cnt > 1: 1 (3)
:         :         :         chg_bank_country_cnt <= 1:
:         :         :         : ....chg_device_cnt > 1: 0 (28)
:         :         :         :         chg_device_cnt <= 1:
:         :         :         :         : ....chg_ip_cnt <= 3: 1 (12/3)
:         :         :         :         :         chg_ip_cnt > 3: 0 (6)
ip_card_mismatch > 0:
: ....chg_seller_cnt <= 1:
:         : ....ip_card_mismatch <= 8:
:         :         : ....chg_email_cnt <= 1: 0 (334/39)
:         :         :         chg_email_cnt > 1:
:         :         :         : ....chg_ip_cnt <= 4: 0 (63/11)
:         :         :         :         chg_ip_cnt > 4: 1 (5)
:         :         :         :         ip_card_mismatch > 8:
:         :         :         :         : ....chg_email_cnt > 3: 0 (29)
:         :         :         :         :         chg_email_cnt <= 3:
:         :         :         :         :         : ....chg_ip_cnt > 2: 0 (10)
:         :         :         :         :         :         chg_ip_cnt <= 2:
:         :         :         :         :         :         : ....bank_fail_cnt <= 5: 1 (16/2)
:         :         :         :         :         :         :         bank_fail_cnt > 5: 0 (7/1)
chg_seller_cnt > 1:
: ....chg_device_cnt > 4: 0 (11)
:         chg_device_cnt <= 4:
:         : ....chg_email_cnt > 4: 0 (25/2)
:         :         chg_email_cnt <= 4:
:         :         : ....pay_succ_cnt > 3: 1 (11/1)
:         :         :         pay_succ_cnt <= 3:
:         :         :         : ....pay_succ_cnt <= 0:
:         :         :         :         : ....amt <= 412.1: 1 (87/35)
:         :         :         :         :         : amt > 412.1: 0 (31/7)
:         :         :         :         :         : pay_succ_cnt > 0:
:         :         :         :         :         :         : ....chg_card_cnt > 2: 1 (4/1)
:         :         :         :         :         :         :         chg_card_cnt <= 2:
:         :         :         :         :         :         :         : ....ip_card_mismatch > 1: 0 (22/1)
:         :         :         :         :         :         :         :         ip_card_mismatch <= 1:
:         :         :         :         :         :         :         :         : ....chg_card_cnt <= 1: 1 (4)
:         :         :         :         :         :         :         :         :         chg_card_cnt > 1: 0 (4)

```

同时，还会输出模型在训练集上的错误率以及变量的贡献情况（见代码清单 10-6）。

#### 代码清单 10-6

Evaluation on training data (8208 cases):

Decision Tree

```
Size      Errors
```

```
37 522( 6.4%) <<
```

```
Attribute usage:
```

```
100.00%  chg_card_cnt
99.38%   ip_card_mismatch
95.66%   chg_seller_cnt
19.83%   pay_succ_cnt
14.08%   chg_email_cnt
11.99%   amt
5.17%    chg_ip_cnt
2.98%    chg_device_cnt
0.62%    bank_fail_cnt
0.60%    chg_bank_country_cnt
```

最后, 还会输出树形结构 (见图 10-15), 从图中可以看出共生成了 37 个子节点, 柱状图代表黑样本率, 可以根据业务经验进行人为剪枝。

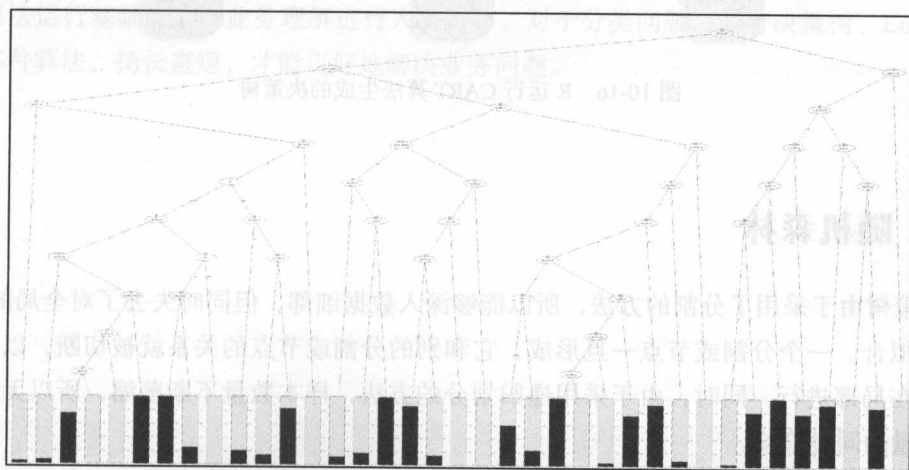


图 10-15 R 运行 CHAID 算法生成的决策树

代码清单 10-7 是用 R 实现 CART 算法的过程, 用到了 `rpart` 包和 CP 剪枝。不过从图 10-16 的树状图来看, 效果并不理想, 碰到这种情况, 可以更换其他算法, 或者增加有区分度的变量。

代码清单 10-7

```
library(rpart)
library(rpart.plot)
library(rattle)
#cart
tree01=rpart(black_flg ~.,data = step01,method = 'class',
```

```
control = rpart.control(minsplit = 1,minbucket = 1,maxdepth = 10))
tree01$cpstable
# 做图
rpart.plot(tree01, branch=1, type=1, extra=105);
fancyRpartPlot(tree01)
asRules(tree01)
```

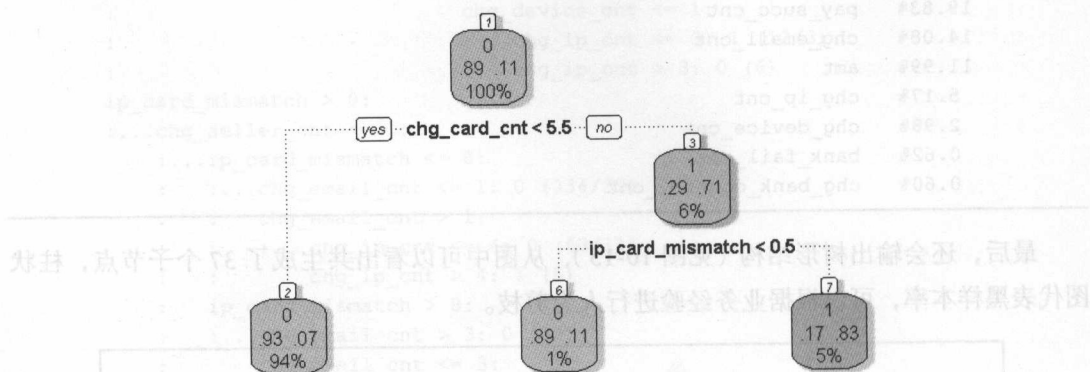


图 10-16 R 运行 CART 算法生成的决策树

## 10.4 随机森林

决策树由于采用了分割的方法，所以能够深入数据细部，但同时失去了对全局的把握，容易过拟合。一个分割或节点一旦形成，它和别的分割或节点的关系就被切断，以后的挖掘只能在局部进行。同时，由于采用递归划分的方法，样本数量不断萎缩，所以无法支持对多变量的同时检验。

由此，产生了随机森林这一算法。顾名思义，森林是由很多树组成的，随机则是指这些树彼此独立没有关联，随机森林就是每次在数据集中对样本（行）和变量（列）分别进行随机抽样，构建出若干棵（通常在 1000 棵左右）决策树，最终组合使用每棵树形成的规则。对于评分或预测来说，每棵树可以看成是一个精通特定领域的专家，针对每个客户的评分是所有专家综合打分的结果，相对单棵决策树来说更加准确、无偏。

随机森林算法具有很多优点，例如能够处理很高维度的数据、训练速度快、实现简单、可以针对不同的数据集并行建模等，目前被广泛应用于信用评分和反欺诈等数据来源广、数据量大、响应率低的场景。关于随机森林的实现过程，可以在决策树构建过程中增加随机抽样和组合打分步骤，建立评分模型，不再详述。

## 10.5 本章小结

以决策树为代表的机器学习算法，本质是针对大量历史数据建立分类模型，基于对“好”样本和“坏”样本的对比，从大量历史案例中总结提炼出“专家”知识，每棵决策树可以看成是一个特定领域的专家。同时，决策树模型具有极好的可解释性，目前被广泛应用于信用、欺诈等各类评分模型中。

与 Logistic 回归相比：① Logistic 回归能够更好地把握数据整体结构，决策树则对局部结构的分析更深入，可以获得局部最优解，因此更像“专家”，非常适合于数据来源广、结构复杂的场景；② Logistic 回归擅长分析线性关系，而决策树则擅长分析非线性关系；③相比 Logistic 回归，决策树对极端值的容忍度更高。由于决策树遵循递归划分的分析过程，后面的分析挖掘只能在前面划分好的节点中进行，节点间的关系无法体现，组合多棵决策树的随机森林算法则能很好地解决这个问题，兼顾局部和整体，得到越来越多的应用。当然，算法是固定的，真正决定模型效果的，还是既符合业务常理又具有区分度的变量，并在算法运行基础上结合业务理解进行人工剪枝。对于分类问题，结合决策树、Logistic 回归等多种算法，扬长避短，才能更好地解决业务问题。

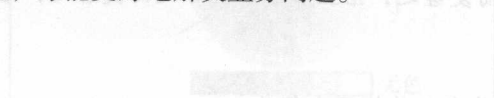


图 10-1 一个标准色块



## 数据可视化，是分析更是设计

言治骨角者，既切之而复磋之；治玉石者，既琢之而复磨之，治之已精，而益求其精也。

——朱熹

格调、和谐、风度和节奏之美均源自简约。

——柏拉图

当前，“用数据说话”已成为职场时尚，因此，辛苦进行的分析，也应该把结论彻彻底底地展现出来。数据和文字是抽象的，图形却是具体的，正所谓“能用图就不用表，能用表就不用文字”，因此制作专业、美观的图形，就成为数据分析人员的必备技能。好的图形或分析报告，应该既直观易懂又不失专业性，数据跃然于纸上，分析一语中的，使决策者醍醐灌顶，这样才能让观者信服。可视化工具虽然可以帮助我们减少制作成本，但工具之外，更需要设计思路和对美的追求。

本章根据笔者的实际经验，介绍图表和报告制作的一些经典规则，并结合两种个性化图形的制作方法，抛砖引玉，当 Excel、Tableau 等可视化工具的默认功能不支持时，应该怎样充分利用免费资源，体现出自己的个性化。

### 11.1 数据演示之道

图表和报告是展示数据分析成果的平台。可是，工作中面目可憎的分析报告随处可见，粗糙错用的图表让人对数据充满质疑，似乎精美专业的图表和报告只应该存在于商业期刊

和顶级咨询公司中。实际上，任何数据分析人员都应该追求制作外观精美简洁、内容严谨专业的图表和报告，因为设计合理的图表不仅可以将分析结论和关键信息直观易懂地展示出来，帮助决策，更可提高数据的可信度。

具体来说，一份好的数据分析报告应该形、色、声俱佳，赋予数据美和洞察力。下面介绍一些规则，让菜鸟做出来的报告也能具备专业水准。

### 11.1.1 好“色”之图

#### 1. 识色相

颜色是数据报告的重要元素，合理用色能够吸引读者的注意、引导读者的阅读，因此首先要知道如何区分色相。图 11-1 是一个标准的色盘，通过它可以认识颜色之间的关系。

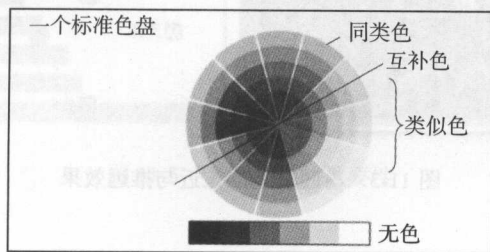


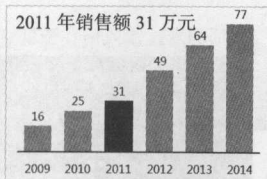
图 11-1 一个标准色盘

#### 2. 能节制

面对丰富的色相，要懂得节制。颜色太多，会让读者的目光迷失，抓不到重点。一般来说，一份报告或图表的色相不宜超过两个。图 11-2 是笔者总结的四种较具代表性的用色风格。

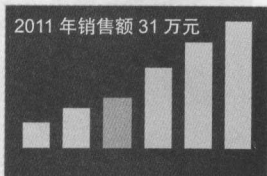
##### 同类色组合

一种颜色、深浅不同  
整体感觉：协调统一、专业



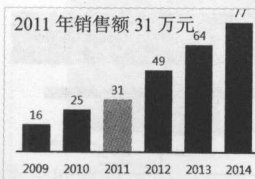
##### 类似色组合

相邻颜色  
突出文字



##### 互补色组合

对立颜色  
掌握深浅才不会突兀



##### 无色彩 + 1 彩色

黑白灰打底  
1 彩色贯穿  
强调

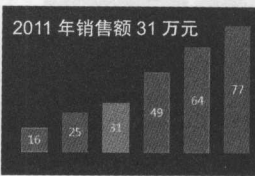


图 11-2 四种代表性的用色风格

### 3. 知冷暖

颜色有冷暖。绿蓝紫等冷色会将距离推远，适合做背景；红橙黄等暖色会拉近距离，适合做前景，突出信息。图 11-3 所示的梵高的这幅图，左图中蓝色的天空看起来很高远，充满了立体感，同样，右图中的橙色将狮子座和巨蟹座突出出来。

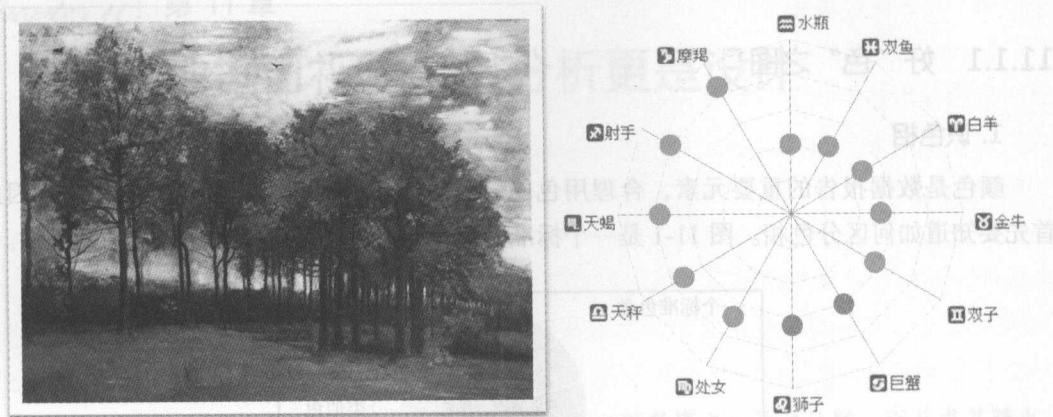


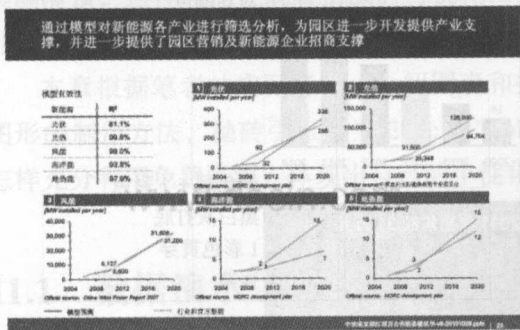
图 11-3 冷暖颜色的拉近与推远效果

## 11.1.2 版式有形

### 1. 风格明确

报告有很多种风格，有的以线条简单勾勒，有的用颜色浓妆艳抹，有的深沉凝重，有的简单清新……无论采用哪种，最好是一种风格贯穿始终，不宜混搭。图 11-4 展示的是两种典型的风格。

清新的罗兰贝风格：清新、明亮、线条



浓郁的麦肯锡风格：浓郁、深沉、底色

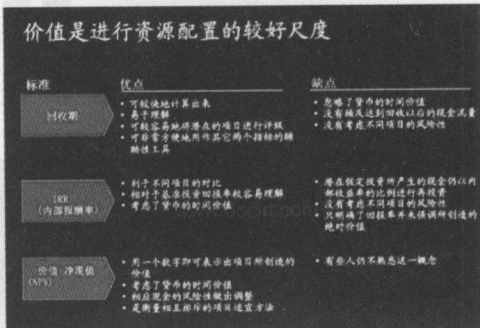


图 11-4 两种典型的风格

## 2. 文字协调

在文字编排上报告和文章一样，通篇应该使用一种主要字体。标题、正文、注释等部分各自保持一致的字号；行距、字间距松紧适度。图 11-5 所示的两张图，右边的图看起来整洁多了吧。



图 11-5 文字协调的重要性

## 3. 懂得留白

除非详细阐述，不要把大段文字码在一页片子中，空白的地方会给人想象的空间。在摄影技术上，留白是一种意境。图 11-6 所示的两张照片效果如何，不用多说。

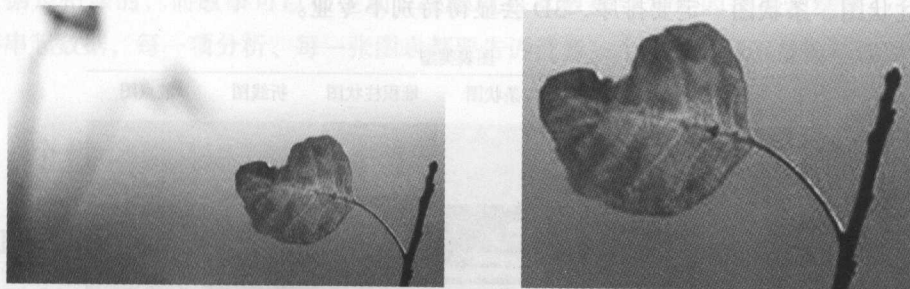


图 11-6 摄影中的留白

## 4. 合理布局

一页幻灯片包含哪些要素是有标准配置的，包括每个要素的位置。如果没有更好的设计，图 11-7 是比较稳妥的布局方式。



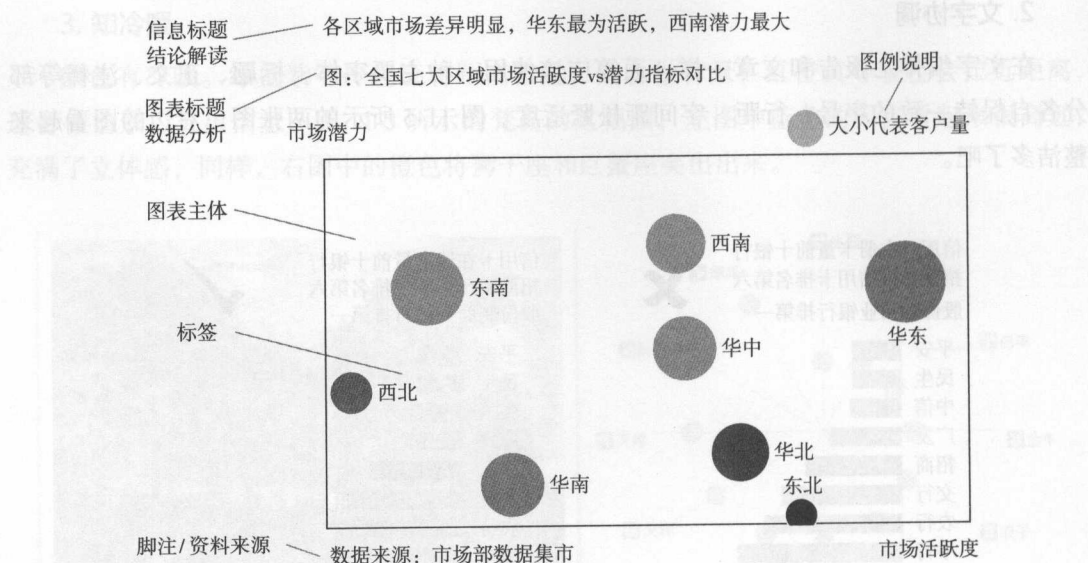


图 11-7 一页幻灯片的布局

### 11.1.3 数据发声

#### 1. 善用图表

选择合适的图表可以增强数据的可读性，让结论一目了然，使用什么图取决于要展示的数据类型以及要传递什么信息。图 11-8 是常见数据类型及其对应的展示方式。需要强调的是，柱状图、条状图一定要排序，3D 会显得特别不专业。

		图表类型					
		饼图	柱状图	条状图	堆积柱状图	折线图	散点图
数据类型	成分比例						
	类别						
	趋势		一定要排序	类别多、标签长时用			
	频率分布						
	关联						

图 11-8 常见数据类型及展示方式

## 2. 大道至简

使用图表的目的是让数据和要传递的信息直观易懂。可很多人偏偏喜欢把图表搞得很复杂，把很多信息挤进一张图表中。记住，一张图片展示的信息越少越好，而不是越多越好，有些不必要的信息宁可不展示，如图 11-9 所示。如果实在有很多信息，与其挤在一起，不如分为多张图片。

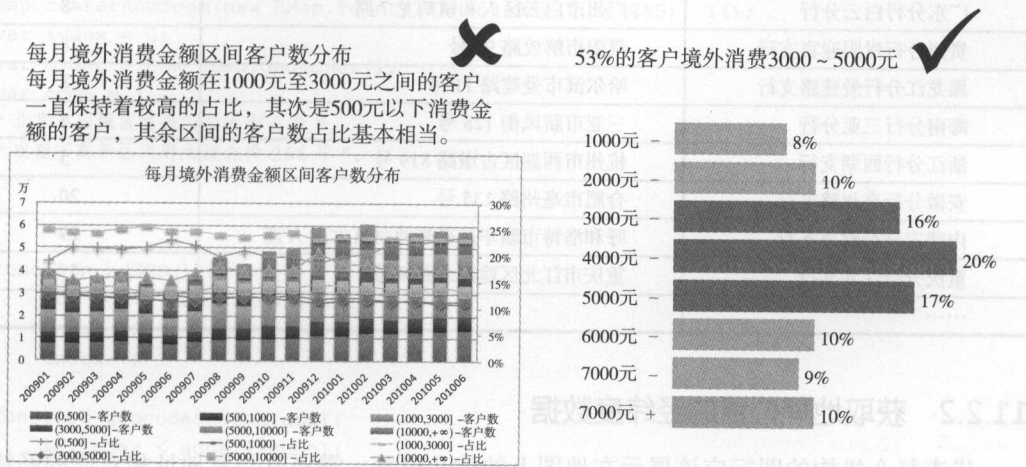


图 11-9 简单的图形更容易理解

## 3. 用故事汇报

数据是枯燥的，而故事可以很生动。故事就是从数据中发现的规律和结论。分析报告要能够串起数据，每一项分析、每一张图表都要告诉读者一个故事，而不是简单地罗列数据。

## 11.2 个性化地图

### 11.2.1 案例背景：存款增长率指标展示

当展示不同地区或地理位置的 KPI 指标时，地图是最佳选择。然而，目前对于大多数可视化工具来说，地图功能都是一项软肋，根本原因是只有极少数公司掌握地图数据，为其他可视化工具提供调用服务。

其实，利用百度 LBS 开放平台可以制作个性化地图，满足分析人员对地图风格、地图颜色、元素显示等的特定需求，根据不同的使用场景和展示需求，打造属于自己的个性化地图。这里结合实例，介绍制作流程。表 11-1 中的数据为某银行全国各分支机构的存款增长率指标，共有机构几千个，这里展示其中的 11 条样例。数据专为本案设计，地址并不

与任何银行一一对应，存款增长率也完全是随机生成，无实际意义。

表 11-1 各机构存款增长率（样例）

分行	地址	增长率 / (%)
北京分行复兴门支行	北京市西城区复兴门内大街 2 号	26
四川分行高新区支行	成都市高新区天府大道北段 966 号	-22
福建分行湖东分行	福州市湖东路 173 号	-20
广东分行白云分行	广州市白云区人和镇鹤龙 7 路	-8
贵州分行贵阳迎宾支行	贵阳市解放路 95 号	28
黑龙江分行爱建路支行	哈尔滨市爱建路 11 号	-28
海南分行三亚分行	三亚市新凤街 128 号	5
浙江分行西湖支行	杭州市西湖区古墩路 819 号	5
安徽分行亳州路支行	合肥市亳州路 135 号	20
内蒙古分行财富支行	呼和浩特市新华东街誉博财富大厦 A 座	29
重庆分行江北支行	重庆市江北区建新北路 9 号	27
.....		

### 11.2.2 获取地理位置的经纬度数据

代表每个机构的图标应该展示在地图上的什么位置，需要有与其地址相对应的经纬度。百度提供了将地址转换为经纬度的接口。当然，地址必须真实有效，通常要包含省（市）、路或者街道以及门牌号。

在代码清单 11-1 中，将地址填入对应位置，就可以获取经纬度。

代码清单 11-1

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf8" />
<meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
<title> 批量地址 </title>
<style type="text/css">
body, html, #allmap {width: 100%;height: 100%;overflow: hidden;margin:0;}
#l-map{height:20%;width:18%;float:left;border-right:2px solid #bcbbc;}
#r-result{height:100%;width:50%;}
</style>
<script type="text/javascript" src="http://api.map.baidu.com/api?v=1.4"></script>
</head>
<body>
<div id="l-map"></div>
<p> &nbsp; </p>
<div id="r-result">
<input type="button" value=" 批量地址解析 " onclick="bdGEO()" />
```

```

<div id="result" style="height:300px;overflow:scroll"></div>
</div>
</body>
</html>
<script type="text/javascript">

// 百度地图 API 功能
var map = new BMap.Map("l-map");
map.centerAndZoom(new BMap.Point(121.483696, 31.244285), 14);
var index = 0;
var myGeo = new BMap.Geocoder();
var adds = [
    "北京市西城区复兴门内大街2号",
    "成都市高新区天府大道北段966号",
    .....
];

function bdGEO(){
    var add = adds[index];
    geocodeSearch(add);
    index++;
}

function geocodeSearch(add){
    if(index < adds.length){
        setTimeout(window.bdGEO, 300);
    }
    myGeo.getPoint(add, function(point){
        if (point) {
            document.getElementById("result").innerHTML += index + "、"
                + add + ":" + point.lng + "," + point.lat + "<br>";
            var marker = new BMap.Marker(new BMap.Point(point.lng, point.lat));
            map.addOverlay(marker);
        }
    }, "北京市");
}
</script>

```

使用浏览器(推荐 Firefox)打开可得到如图 11-10 所示的界面。单击批量解析地址,会生成每个地址的经纬度。将数据复制出来,分列处理,可获取经度和纬度两列数据。

### 11.2.3 定制地图背景和图标

制作地图背景。百度提供了个性地图编辑工具,网址为: <http://developer.baidu>。

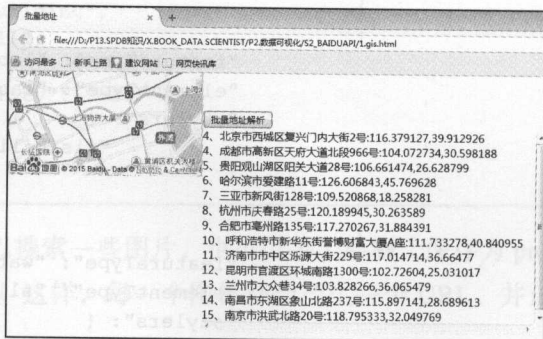


图 11-10 批量获取经纬度



com/map/custom, 在这里可以通过交互的方式自行编辑地图展示样式, 直到满足自己的喜好, 自动生成地图样式 JSON, 通过 JavaScript API 的方法调用生效。

例如, 笔者参照“百度旅游”的风格制作了背景地图, 代码清单 11-2 是该背景地图的 JSON 码。

代码清单 11-2

```
var styleJson = [
  {
    "featureType": "road",
    "elementType": "all",
    "stylers": {
      "visibility": "off"
    }
  },
  {
    "featureType": "land",
    "elementType": "all",
    "stylers": {
      "color": "#1F497D",
      "weight": "0.5",
      "lightness": 1,
      "saturation": 1
    }
  },
  {
    "featureType": "label",
    "elementType": "labels",
    "stylers": {
      "color": "#eeeeee",
      "hue": "#cccccc",
      "weight": "1",
      "lightness": 1,
      "saturation": 1,
      "visibility": "off"
    }
  },
  {
    "featureType": "label",
    "elementType": "labels.icon",
    "stylers": {
      "visibility": "off"
    }
  },
  {
    "featureType": "water",
    "elementType": "all",
    "stylers": {
      "color": "#002960"
    }
  }
]
```

```

    },
    {
      "featureType": "boundary",
      "elementType": "all",
      "stylers": {
        "color": "#556579"
      }
    },
    {
      "featureType": "boundary",
      "elementType": "geometry.stroke",
      "stylers": {
        "weight": "0.1"
      }
    },
    {
      "featureType": "label",
      "elementType": "labels",
      "stylers": {
        "color": "#eeeeee",
        "lightness": 1,
        "saturation": 10
      }
    },
    {
      "featureType": "green",
      "elementType": "all",
      "stylers": {
        "visibility": "off"
      }
    },
    {
      "featureType": "poi",
      "elementType": "all",
      "stylers": {
        "visibility": "off"
      }
    },
    {
      "featureType": "building",
      "elementType": "all",
      "stylers": {
        "visibility": "off"
      }
    }
  ]
}

```

表中的数据指标通过什么图标展示，可以搜索一些图片，也可以手工绘制，保存为 png 图片，将其上传到网上，例如百度个人相册。这样，每一个图标都会产生一个 URL，并记录下来。

### 11.2.4 生成地图

将各个步骤得到的机构名称、经纬度、数据指标及其对应的图标、图标 url、背景地图的 JSON 代码等，放到绘图 html 代码的对应位置，如代码清单 11-3 所示。

代码清单 11-3

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=GB18030" />
  <meta name="viewport" content="initial-scale=1.0, user-scalable=no" />
  <style type="text/css">
    body, html {width: 100%;height: 100%;margin:0;font-family:" 微软雅黑 ";}
    #allmap{width:100%;height:700px;}
    p{margin-left:5px; font-size:14px;}
  </style>
  <script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=Pxr
    d2i8Qnf6mOUTCdHixxxxx">
// 注：该 ak 码隐去了后 5 位，读者可从百度注册获取
</script>
  <title></title>
</head>
<body>
  <div id="allmap"></div>
  <p>* 机构存款增长率指标 缩放查看区域分布
    * 单击图标查看机构信息 </p>
</body>
</html>
<script type="text/javascript">
  var map = new BMap.Map('allmap');
  map.centerAndZoom(new BMap.Point(116.404269,39.916042), 12);

  map.addControl(new BMap.NavigationControl()); // 添加平移缩放控件
  map.addControl(new BMap.ScaleControl()); // 添加比例尺控件
  map.addControl(new BMap.OverviewMapControl()); // 添加缩略地图控件
  map.enableScrollWheelZoom(); // 启用滚轮放大缩小
  map.disable3DBuilding();

... // 此处填入代码清单 11-2 背景地图的 JSON 码

map.setMapStyle({styleJson:styleJson});
var data_info = [
  [114.303936,30.582744," 湖北分行江岸支行,82"],
  [126.640348,45.610758," 黑龙江分行南岗支行,75"],
  .....
];
var opts = {
  width : 250, // 信息窗口宽度
  height: 80, // 信息窗口高度
```

```

        title : " 网点指标: " , // 信息窗口标题
        enableMessage:true // 设置允许信息窗发送短消息
    };

    for(var i=1;i<=90;i++){
        var myIcon = new BMap.Icon("http://e.picphotos.baidu.com/album/
s%3D1100%3Bq%3D90/sign=ffc03a5d8444ebf86971603ee9c9ec5b/c2cec3fdcf0392454228e251
8494a4c27cle25ca.jpg", new BMap.Size(6,6));
        var marker = new BMap.Marker(new BMap.Point(data_info[i][0],data_
info[i][1]),{icon:myIcon}); // 创建标注
        var content = data_info[i][2];
        map.addOverlay(marker); // 将标注添加到地图中
        addClickHandler(content,marker);
    }

    for(var i=91;i<=268;i++){
        var myIcon = new BMap.Icon("http://b.picphotos.baidu.com/album/
s%3D1100%3Bq%3D90/sign=d482849540166d223c77119576133286/241f95cad1c8a786486e4950
6409c93d70cf5023.jpg", new BMap.Size(6,6));
        var marker = new BMap.Marker(new BMap.Point(data_info[i][0],data_i
nfo[i][1]),{icon:myIcon}); // 创建标注
        var content = data_info[i][2];
        map.addOverlay(marker); // 将标注添加到地图中
        addClickHandler(content,marker);
    }

    for(var i=269;i<=536;i++){
        var myIcon = new BMap.Icon("http://a.picphotos.baidu.com/album/
s%3D1100%3Bq%3D90/sign=2e4b2dbbae6eddc422e7b0fa09eb8d8c/5ab5c9ea15ce36d3f586806e
39f33a87e850ble9.jpg", new BMap.Size(6,6));
        var marker = new BMap.Marker(new BMap.Point(data_info[i][0],data_
info[i][1]),{icon:myIcon}); // 创建标注
        var content = data_info[i][2];
        map.addOverlay(marker); // 将标注添加到地图中
        addClickHandler(content,marker);
    }

    for(var i=537;i<=893;i++){
        var myIcon = new BMap.Icon("http://h.picphotos.baidu.com/album/
s%3D1100%3Bq%3D90/sign=3c987f5d8444ebf86971603ee9c9ec5b/c2cec3fdcf0392458170a751
8494a4c27cle25a2.jpg", new BMap.Size(6,6));
        var marker = new BMap.Marker(new BMap.Point(data_info[i][0],data_
info[i][1]),{icon:myIcon}); // 创建标注
        var content = data_info[i][2];
        map.addOverlay(marker); // 将标注添加到地图中
        addClickHandler(content,marker);
    }

    function addClickHandler(content,marker){
        marker.addEventListener("click",function(e){
            openInfo(content,e)
        });
    }

    function openInfo(content,e){
        var p = e.target;

```



```

11.2.4 生图
var point = new BMap.Point(p.getPosition().lng, p.getPosition().lat);
var infoWindow = new BMap.InfoWindow(content,opts); // 创建信息窗口对象
map.openInfoWindow(infoWindow,point); // 开启信息窗口
}
</script>

```

同样，使用浏览器（推荐 Firefox）打开，即可得到属于你的个性化地图，图 11-11 是北京分行辖下各营业网点的指标。这个地图具有缩放功能，可以放大局部区域，对于指标异常（例如资金流出严重）的机构，可单击图标查看更多信息。

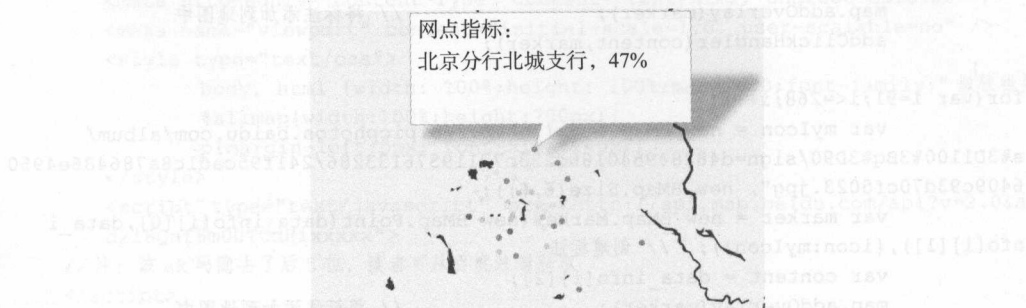


图 11-11 展示机构存款增长率的个性化地图

## 11.3 文本分析

业务过程中会产生大量的文本数据，例如贷款合同、客户投诉、网络信息等，如何对这些文本类型的数据进行分析并发现营销或风险线索，属于另一个分析范畴——NLP。NLP（natural language processing）是人工智能（AI）的一个子领域，其主要应用场景包括中文分词、信息抽取、信息检索、机器翻译、自动摘要、文本分类、舆情分析等。这其中存在很多非常具有挑战性的课题，例如，如何从大段文字中提炼出公司名称（有时还要缩写）等关键词，如何界定每个词语在上下文中的含义，如何定义一句话是褒义还是贬义等。因此，NLP 也成为人工智能中最为困难的问题之一。作为数据分析人员，很多时候只需要知道怎么做文本分析，不必担心复杂的 NLP 算法。这里，介绍最简单实用的词云制作和情感分析。

### 11.3.1 案例：电商的客户评价分析

电商网站都提供商品评价功能，消费者在网上购买商品之后，会针对商品的外观、价格、使用情况等进行评论、点赞或者拍砖。对消费者来说是发泄情感，对电商和商品制造商来说则是获取第一手市场信息的渠道，发现自己产品的不足进行改进，或者发现潜在的

产品创新机会，推出迎合消费者喜好或具有市场前景性的产品。某电商在一款新上市的手机销售3个月后，打算针对差评信息进行一次分析，以及时发现产品或自己服务、物流中存在的不足。表11-2是其中的差评样例，全部差评约1万条。

表 11-2 消费者对一款新上市手机的差评（10 条样例）

客户编号	评论内容
1001	太差，不怎么样，本来对这款手机不抱有什希望，还不如我之前用的 HTC，差太远。先说系统，图标真的是难看死了，你们有设计吗？请问，有吗？再说打电话时，声音大一点，后面板就会震动！！后来，数据网也连不上了
1002	机子刚刚买了两个星期，轻轻一摔就这个样子了，返修审核不过。在当地维修 800 的费用，没有半点优惠。在考虑如何维修中
1003	就是系统有点傻，再就是电量问题，而且根据目前各大测试，尤其是科技美学的测试，除了相机，其他的都被 pk 掉了，有点伤心。不懂为什么还这么贵，有毛病……为什么感觉被阉割的感觉。很烦的……
1004	手机刚收到有些小瑕疵，屏幕有点轻微划痕，联系客服不给退换货。拍照摄像头发热严重，电池电量一般
1005	收到手机开机后，屏幕左上方显示出一道至少 5cm 的裂痕，我也是醉了！收到手机开机后，屏幕左上方显示出一道至少 5cm 的裂痕，我也是醉了！收到手机开机后，屏幕左上方显示出一道至少 5cm 的裂痕，我也是醉了！重要的事情说三遍
1006	到货就看到手机有问题，而且通话拍摄都发烫，用得都担心，退回去不给退。太坑了，怀疑是不是发个二手货的。真气人，买个这样的手机！
1007	暂不说实际拍照能力到底有没有宣传的那么牛叉，刚用一个月，完全日常使用环境，摄像头进这么多灰，这做工品控也是没得说了，去售后直接给打开清理了下
1008	老实说吧，电板不耐用，触屏有时候不灵，手机外观完美，大小合适。吐槽一下插卡模式，还需要借助针才能打开，要是针丢了呢？不打开了？总体来说有点小失望
1009	刚买了一个月时间，昨天晚上发现双摄像头处的玻璃开裂了，说良心话，自己还是很爱惜着用的，很郁闷。要自费去维修，感觉像吃了只苍蝇，很恶心。从裂缝处明显能看到，这是沿着两个摄像头方向裂的，是两个摄像头离太近削弱了此处的玻璃强度。明显是质量缺陷，却要消费者买单。这样的质量 ** 能走多远，这样的售后 ** 又能活多久
1010	真的垃圾！网络只能用移动 4G，连 2G、3G 都不能用，去偏僻的地方没 4G 完全用不了！就一部移动定制机！和移动合作骗钱的玩意。新机都卡得不行，垃圾。拍照饱和度高得吓人，失真，千万别买
.....	

注：评论信息仅为介绍分析方法，本书不持任何观点。

### 11.3.2 分词

分词是文本分析的第一步。通过分词，将连续的文字划分为具有实际含义的若干个词语，目前针对中文的分词都基于词库实现，因此分词的效果好坏取决于词库。网上有很多开源工具，可在词库基础上包装一个操作界面，如图 11-12 所示。

选择表 11-2 的差评文件，指定一种编码方式（具体取决于文字），单击“提交”按钮，将生成两个结果。

其一为分词，如图 11-13 所示。通过“/”将词语分割开，“/”后注明该词的词性，例

如名词、动词、形容词等。将文字导入 SAS 等分析工具，可以进行深入分析。

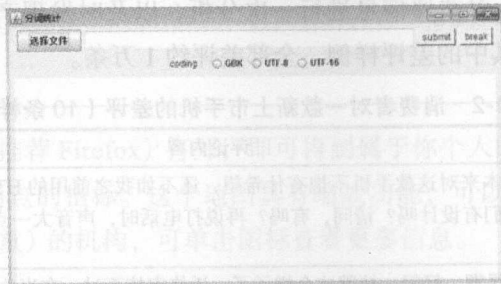


图 11-12 一个分词工具的界面



图 11-13 分词结果 (部分截图)

其二为词频统计，结果如图 11-14 所示。在分词基础上统计了每个词语的频数。

24小时	36小时	48小时	72小时	96小时	120小时	144小时	168小时	192小时	216小时	240小时	264小时	288小时	312小时	336小时	360小时	384小时	408小时	432小时	456小时	480小时	504小时	528小时	552小时	576小时	600小时	624小时	648小时	672小时	696小时	720小时	744小时	768小时	792小时	816小时	840小时	864小时	888小时	912小时	936小时	960小时	984小时	1008小时	1032小时	1056小时	1080小时	1104小时	1128小时	1152小时	1176小时	1200小时	1224小时	1248小时	1272小时	1296小时	1320小时	1344小时	1368小时	1392小时	1416小时	1440小时	1464小时	1488小时	1512小时	1536小时	1560小时	1584小时	1608小时	1632小时	1656小时	1680小时	1704小时	1728小时	1752小时	1776小时	1800小时	1824小时	1848小时	1872小时	1896小时	1920小时	1944小时	1968小时	1992小时	2016小时	2040小时	2064小时	2088小时	2112小时	2136小时	2160小时	2184小时	2208小时	2232小时	2256小时	2280小时	2304小时	2328小时	2352小时	2376小时	2400小时	2424小时	2448小时	2472小时	2496小时	2520小时	2544小时	2568小时	2592小时	2616小时	2640小时	2664小时	2688小时	2712小时	2736小时	2760小时	2784小时	2808小时	2832小时	2856小时	2880小时	2904小时	2928小时	2952小时	2976小时	3000小时	3024小时	3048小时	3072小时	3096小时	3120小时	3144小时	3168小时	3192小时	3216小时	3240小时	3264小时	3288小时	3312小时	3336小时	3360小时	3384小时	3408小时	3432小时	3456小时	3480小时	3504小时	3528小时	3552小时	3576小时	3600小时	3624小时	3648小时	3672小时	3696小时	3720小时	3744小时	3768小时	3792小时	3816小时	3840小时	3864小时	3888小时	3912小时	3936小时	3960小时	3984小时	4008小时	4032小时	4056小时	4080小时	4104小时	4128小时	4152小时	4176小时	4200小时	4224小时	4248小时	4272小时	4296小时	4320小时	4344小时	4368小时	4392小时	4416小时	4440小时	4464小时	4488小时	4512小时	4536小时	4560小时	4584小时	4608小时	4632小时	4656小时	4680小时	4704小时	4728小时	4752小时	4776小时	4800小时	4824小时	4848小时	4872小时	4896小时	4920小时	4944小时	4968小时	4992小时	5016小时	5040小时	5064小时	5088小时	5112小时	5136小时	5160小时	5184小时	5208小时	5232小时	5256小时	5280小时	5304小时	5328小时	5352小时	5376小时	5400小时	5424小时	5448小时	5472小时	5496小时	5520小时	5544小时	5568小时	5592小时	5616小时	5640小时	5664小时	5688小时	5712小时	5736小时	5760小时	5784小时	5808小时	5832小时	5856小时	5880小时	5904小时	5928小时	5952小时	5976小时	6000小时	6024小时	6048小时	6072小时	6096小时	6120小时	6144小时	6168小时	6192小时	6216小时	6240小时	6264小时	6288小时	6312小时	6336小时	6360小时	6384小时	6408小时	6432小时	6456小时	6480小时	6504小时	6528小时	6552小时	6576小时	6600小时	6624小时	6648小时	6672小时	6696小时	6720小时	6744小时	6768小时	6792小时	6816小时	6840小时	6864小时	6888小时	6912小时	6936小时	6960小时	6984小时	7008小时	7032小时	7056小时	7080小时	7104小时	7128小时	7152小时	7176小时	7200小时	7224小时	7248小时	7272小时	7296小时	7320小时	7344小时	7368小时	7392小时	7416小时	7440小时	7464小时	7488小时	7512小时	7536小时	7560小时	7584小时	7608小时	7632小时	7656小时	7680小时	7704小时	7728小时	7752小时	7776小时	7800小时	7824小时	7848小时	7872小时	7896小时	7920小时	7944小时	7968小时	7992小时	8016小时	8040小时	8064小时	8088小时	8112小时	8136小时	8160小时	8184小时	8208小时	8232小时	8256小时	8280小时	8304小时	8328小时	8352小时	8376小时	8400小时	8424小时	8448小时	8472小时	8496小时	8520小时	8544小时	8568小时	8592小时	8616小时	8640小时	8664小时	8688小时	8712小时	8736小时	8760小时	8784小时	8808小时	8832小时	8856小时	8880小时	8904小时	8928小时	8952小时	8976小时	9000小时																																																																																																																																																																																																																																																																																																																																																																													
讲话	1次	1个	1要	3大	2划值	1回去	1800	1裂现	3买																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																											</

图 11-14 词频 (部分截图)

### 11.3.3 词云制作

将全部差评的词频导入 SAS 或 Excel 中，进行过滤，剔除“那”“着”等没有信息含量的词语，得到差评中的高频词汇。其中关于产品内容的词汇尤其应该引起关注，因为这些





表 11-3 词语的情感分(样例)

正负面	词语	情感分
负面	苍蝇	-10
负面	差	-10
负面	恶心	-10
负面	二手货	-10
负面	死	-10
负面	垃圾	-9
负面	骗	-9
负面	气人	-8
负面	伤心	-8
负面	失望	-8
负面	郁闷	-8
负面	坑	-7
负面	削弱	-7
负面	烦	-6
负面	毛病	-6
负面	傻	-6
负面	缺陷	-5
负面	问题	-4
负面	不行	-3
负面	不灵	-3
负面	失真	-3
负面	瑕疵	-2
中性	感觉	0
中性	联系	0
中性	显示	0
正面	牛	6
正面	完美	10
.....		

对照词语的情感分字典,将每个消费者评价中出现的负面词语进行加总,就可以得到每个人的情感评分,略感不满还是严重抗议,得以量化。表 11-4 是这 10 个样例消费者的情感评分,同样,只针对负面词语,满满的负能量。

表 11-4 消费者的情感评分(10 条样例)

客户编号	负面词 1	负面词 2	负面词 3	负面词 4	负面词 5	负面词 6	负面词 7	负面词 8	负面词 9	情感评分
1001	差	死								-20
1002	返修	半点								-7
1003	傻	问题	伤心	毛病	阉割	烦				-39

(续)

客户编号	负面词 1	负面词 2	负面词 3	负面词 4	负面词 5	负面词 6	负面词 7	负面词 8	负面词 9	情感评分
1004	瑕疵	划痕	退换	发热	严重	一般				-33
1005	裂痕	醉								-8
1006	问题	烫	担心	退	坑	二手货		气人		-46
1007										0
1008	不灵	吐槽	垃圾	死	失望					-36
1009	开裂	郁闷	自费	维修	苍蝇	恶心	裂缝	削弱	缺陷	-60
1010	垃圾	骗	卡	不行	垃圾	吓人	失真			-44
.....										

## 11.4 本章小结

本章围绕数据可视化主题介绍了图表和报告制作过程中的一些规则，让分析结论看起来更加专业、可信。特别强调的是，决定数据可视化效果的不是工具，而是设计，设计最合适的图形、表格呈现分析结论，或者想要告诉别人的信息。优秀的报告应该具备两个特质：其一，言简意赅陈述自己的主题；其二，快速准确地突出观众关注的核心问题。这样的报告会让人听起来很轻松，而且容易产生共鸣，这两点也是报告演示的原则。

第三部分 *Part 3*应用篇  
(Applications)

成大事若烹小鲜，做大事必重细节。天下难事，  
必作于易；天下大事，必作于细。

——老子《道德经》

小事成就大事，细节成就完美。

——戴维·帕卡德（惠普创始人）

好在“空间换时间”这个数据应用领域的“优良传统”仍然适用于此处。我们可以事先设计出所有可能被应用于生产系统的属性，再通过定时任务事先计算出所有属性的值，并将这些属性值按照特定的规则存储于生产环境数据库中（大量的存储空间）。构建在生产数据库之上的生产系统就可以通过“点”查询（少量的时间）的方式对业务请求进行回应。标签系统，就是符合这种理念的数据驱动的应用系统。

## 12.1 认识标签系统

这里所说的标签是指数据领域的标签，也可以称为数据标签。数据标签是对实体

数据由生产系统产生后，经过数据 ETL、信息统一、格式规整等过程进入分析环境，这是数据经历的第一次价值提升。在分析环境中，数据工程师对数据进行分析挖掘，从数据中进一步提炼出“价值”，这是数据脱离生产环境后的第二次价值提升。要形成完整的数据闭环，则需要数据的第三次价值提升，即数据重新应用于生产系统，“从生产中来，到生产中去”。

过去，数据应用于生产系统一般局限于周期性的、批量性的应用，比如从大量信用卡持卡人中，通过专家规则或者数学模型筛选出一部分客户，并对这些客户进行批量营销。整个过程手工操作占了绝大部分，这并不是真正意义上的数据驱动的应用系统。

真正的数据驱动的应用系统，应该是立足于数据，使用各种自动化手段，让数据“自动”流转于各个环节，并具有灵活的可配置性（通过配置满足多种应用场景）和交互性（有 Web 界面可供查看，而不全是后台处理）。



## 标签系统

闲居少邻并，草径入荒园。鸟宿池边树，僧敲月下门。过桥分野色，移石动云根。暂去还来此，幽期不负言。

——贾岛《题李凝幽居》

管理是一种器官，是赋予机构以生命的、能动的、动态的器官。没有机构，就不会有管理。但是，如果没有管理，那也就只会有一群乌合之众，而不会有一个机构。

——彼得·德鲁克

从批量处理的分析环境到偏重“点”处理（逐条）的应用系统，着重要解决的是数据处理的时效问题。当数据应用系统接收到业务请求时，系统需要快速得出结果，做出反馈。因此，效率问题是数据应用于生产面临的首要问题。

好在“空间换时间”这个数据应用领域的“优良传统”仍然适用于此处。我们可以事先设计出所有可能被应用于生产系统的属性，再通过定时任务事先计算出所有属性的值，并将这些属性值按照特定的规则存储于生产环境数据库中（大量的存储空间），构建在生产数据库之上的生产系统就可以通过“点”查询（少量的时间）的方式对业务请求进行回应。标签系统，就是符合这种理念的数据驱动的应用系统。

### 12.1 认识标签系统

这里所说的标签是指数据领域的标签，也可以称为数据标签。数据标签是对实体

(entity) 属性的描述, 标签的值标记了实体的一个信息。例如, 对于一个信用卡持卡人, 性别是其一个标签, “女性”是这个标签的值, 它标记了该持卡人的性别信息。

标签系统是根据一定的规则, 计算和存储标签的集合, 它按照既定的逻辑对标签进行分类管理, 并根据规则进行标签值的计算和更新。

实体的属性多种多样, 因此对这些属性进行数据表述的标签同样多种多样。有些标签描述的是实体的原始属性, 比如持卡人的性别、出生日期、出生地等; 有些标签则是对实体历史属性的描述, 如持卡人过去 3 个月的消费金额、持卡人近 3 个月的刷卡次数等; 还有些标签是对实体未来属性的预测, 如持卡人在未来 6 个月内销卡的概率、持卡人在未来 1 个月内进行账单分期的意愿等。

既然标签是对实体属性的描述, 而在数据库中实体的属性即对应着数据表的字段, 那我们为什么还需要标签呢? 表 12-1 总结了使用标签的原因。

表 12-1 使用标签的原因

序 号	原 因	说 明
1	解决重复计算问题	一次计算, 多次访问
2	解决查询时效问题	存储计算结果, 无需计算
3	降低数据使用成本	使数据使用不存在技术门槛
4	为其他应用提供支持	可以作为数据挖掘、数据营销的底层支持

仍以上述持卡人为例, 每个持卡人的历史交易信息都存储在银行的数据库中, 当要查看某个持卡人近 3 个月的消费金额时, 你需要撰写一段 SQL 脚本并执行它, 然后数据库通过计算返回该持卡人近 3 个月的交易金额, 这个计算过程可能是一种资源浪费, 因为你的同事可能已经多次以相同的方式查询过持卡人“近 3 个月交易金额”这个指标了。而通过设置“近 3 个月交易金额”标签, 通过周期性预先计算, 就可以为所有查询该指标的用户直接提供标签值, 避免了重复计算, 同时提高了查询效率。

标签解决的另一个问题是数据使用成本问题, 在很多企业中, 一些非技术部门人员需要查看用户的一些信息。大多数情况下, 他们没有直接查询数据库的权限(也许权限不是问题, 但他们不具备自行查询数据库的技术能力), 因此需要将需求提交给数据工程师代为处理。预计算的标签解决了这些问题, 让计算和技术对用户透明, 降低了数据使用门槛。

另外, 标签还是众多数据应用的基础, 例如, 作为数据建模时的输入, 可为规则引擎提供支持等, 这是数据从分析环境走向生产环境的基础, 也是本书将标签系统作为“应用篇”首章的原因所在。

## 12.2 标签系统的设计

### 12.2.1 标签系统的层次结构

标签系统是一系列标签的集合，为了便于管理，这里借用数据仓库理论中的一个术语：主题。将同属于一个类别的标签归为一个主题，并且在同一个主题下面的标签，其存储和访问方式遵循相同的规则。这样，一个标签系统就可以在逻辑上分为多个标签主题，比如在信用卡系统中根据标签描述的对象，可以将标签系统分为客户主题、卡片主题等。

每个标签主题需要指定一个主题 ID，用于标识该标签主题下的实体区分方式，一般区分数据库的主键或者 Hbase 表中的 RowKey。例如，客户主题的主题 ID 为客户号，卡片主题的主题 ID 为卡号。

标签主题可以进一步分为多个类别，可以根据标签数据的来源进行划分，例如客户主题的标签可以细分为基础标签、行为标签等。其中基础标签描述的是客户的基本信息，如性别、出生日期、国籍、工作单位、年收入等，这些数据不需要二次加工；行为标签用于描述客户的历史行为，如近 1 个月投资金额、近 1 个月投资笔数等，这些标签需要对数据进行二次加工。

最终的标签位于标签类别之下，例如，基础类标签下包含性别、出生日期、国籍、工作单位等。如果将整个标签系统看成一棵树，则标签是树上的叶子节点。图 12-1 展示了标签系统的层次结构示意图，图中每个主题都包含基础类、行为类和衍生类三种类别的标签。层次化的标签系统更容易管理和扩充，并且在实现上也比较容易。

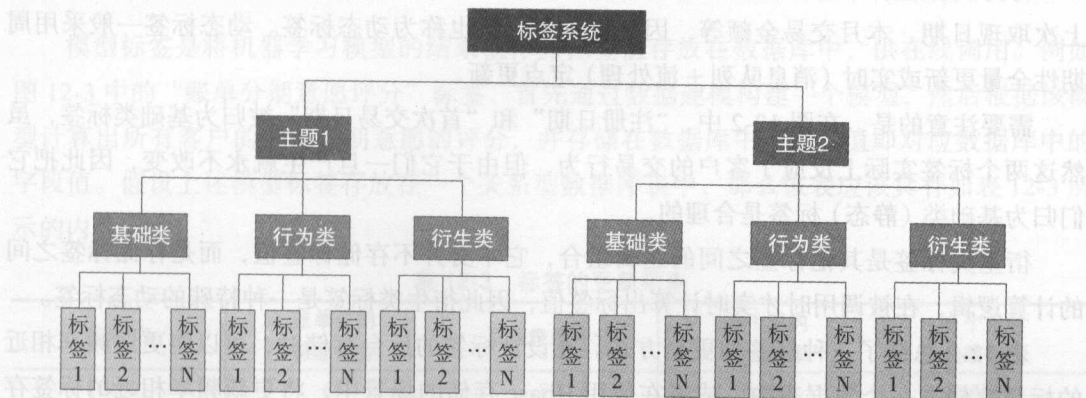


图 12-1 标签系统的层次结构示意图

### 12.2.2 标签系统的更新规则

标签的层次结构允许针对不同的标签类型使用不同的存储和更新方式，标签类型作为

标签的上层根节点，决定标签的存储和更新方式。以信用卡系统中的客户主题为例，客户主题下包含基础类、行为类和衍生类三种标签类别，如图 12-2 所示。

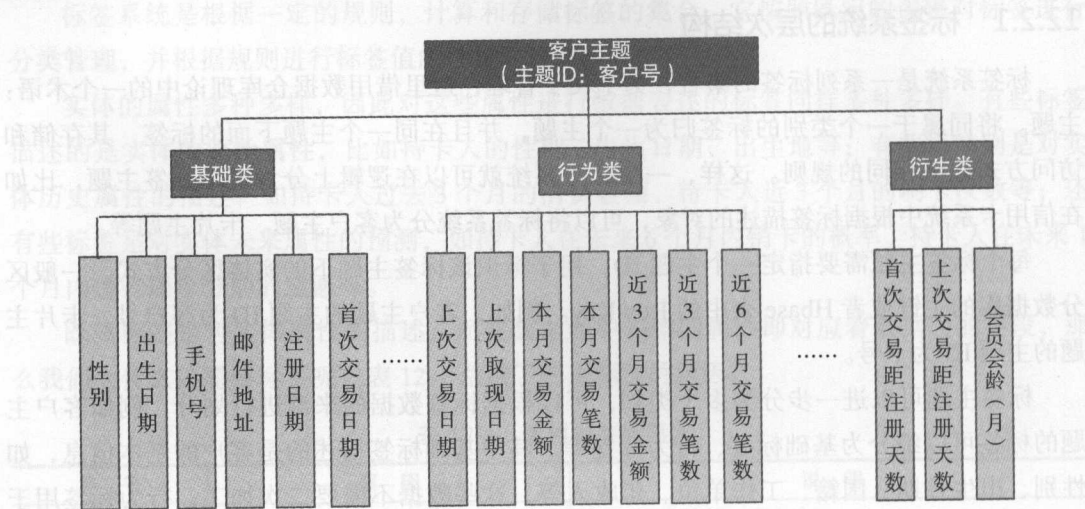


图 12-2 客户主题下的部分标签

图 12-2 中的标签，客户主题分为三个类别：基础类、行为类和衍生类。基础类标签的值一般固定不变，或者更新周期很长，如性别、出生日期、注册日期、首次交易日期等，因此基础类标签可以称为静态标签。显然，针对基础类标签，可以使用增量更新的方式刷新标签值，每次仅更新值改变的标签或者插入新增加的客户端基础标签即可。

行为类标签标记客户的历史行为，这种标签总是处于变化之中，例如上次交易日期、上次取现日期、本月交易金额等，因此行为类标签也称为动态标签。动态标签一般采用周期性全量更新或实时（消息队列 + 流处理）定点更新。

需要注意的是，在图 12-2 中，“注册日期”和“首次交易日期”被归为基础类标签，虽然这两个标签实际上反应了客户的交易行为，但由于它们一旦产生就永不改变，因此把它们归为基础类（静态）标签是合理的。

衍生类标签是其他标签之间的逻辑组合，它本身并不存储标签值，而是存储标签之间的计算逻辑，在被调用时才实时计算出标签值，因此衍生类标签是一种特殊的动态标签。

表 12-2 总结了三种标签的更新方式。在设计标签的后台存储时，可以将更新频率相近的标签存储在一个数据表中，或者在使用 Hbase 存储的场景中，将更新频率相近的标签存放在相同的列族（column family）中。

表 12-2 标签的更新频率

类 型	更新频率	更新方式	存储方式
静态标签	低	增量更新	关系型数据库



(续)

类 型	更新频率	更新方式	存储方式
动态标签	高	全量更新或实时更新	Hbase
虚拟标签	无须更新		关系型数据库

### 12.2.3 机器学习模型转化为标签

机器学习模型在建模阶段需要大量的计算，因此建模过程大多是通过离线计算完成的。为了完成机器模型的在线引用，可以将机器学习模型转换为标签，供在线使用。

仍以上述信用卡系统标签为例，我们在客户主题下创建一个新的标签类型：模型标签，如图 12-3 所示。

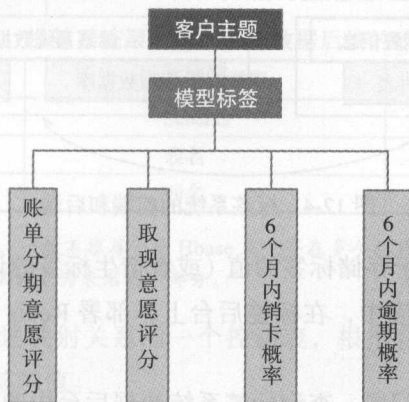


图 12-3 模型标签

模型标签是将机器学习模型的结果值作为标签值存放在数据库中，供在线调用。例如图 12-3 中的“账单分期意愿评分”标签，首先通过数据建模构建一个模型，然后根据该模型计算出所有客户的账单分期意愿的评分，并存储在数据库中，标签值即对应数据库中的字段值。假设上述模型标签存放在一个关系型数据库表中，那么该表应该具有如表 12-3 所示的内容。

表 12-3 标签的更新频率

客户 ID	账单分期意愿评分	取现意愿评分	6个月内销卡概率	6个月内逾期概率
100001	65	80	0.001	0.0001
100002	90	86	0.77	0.58
100003	40	20	0.23	0.11

通过将模型值映射为标签值，我们为机器学习模型的在线应用提供了一种简单易行的途径。

### 12.3 标签系统的实现

如上所述，我们明确了标签系统的设计理念。不难发现，标签系统是一个“数据密集型”的系统，数据后台设计是标签系统的重头戏。数据后台负责标签值的存储和更新，在此之上，还需要一个前端管理页面，用于查询和设置标签名称与后台的映射关系。图 12-4 所示的为标签系统的前后端示意图。

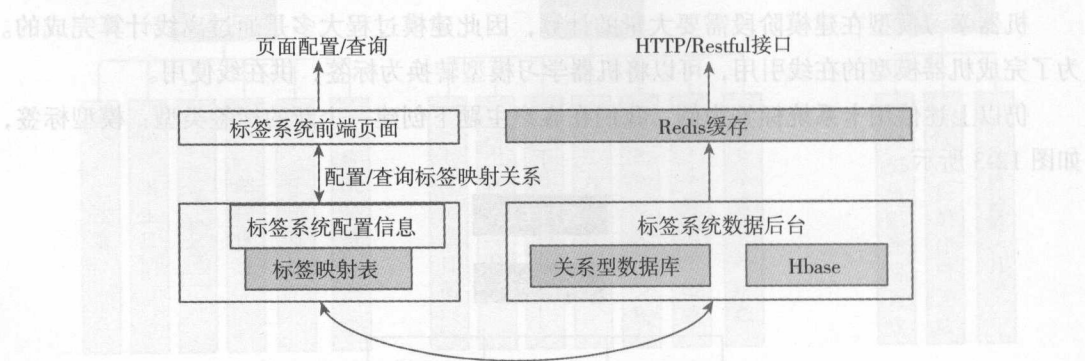


图 12-4 标签系统的前端和后端

标签系统的数据后台用于存储标签的值（或者衍生标签的计算公式），在物理上存储在关系型数据库表或者 Hbase 表中，在标签后台上，部署 Redis 缓存，用于为 HTTP 接口提供快速读取服务。

标签系统前端页面用于配置、查询标签系统数据后台中的标签，通过标签映射表将标签名称与数据后台中的标签值存储位置关联起来。通过标签映射表中存储的信息，对外接口就可以通过标签名称自动找到标签的值。

标签系统可以提供多种服务，如标签值页面查询服务、HTTP 在线接口查询服务、批量标签支持等，如表 12-4 所示。

表 12-4 标签的更新频率

	输 入	输 出
标签值页面查询	标签名称、主题 ID 值	在页面上展示标签值
标签值接口查询	标签名称、主题 ID 值	包装成 Json 格式的结果
批量标签支持	主题名称	主题中所有的标签值

用户可以通过“标签值页面查询”页面，输入需要查询的标签名称和主题 ID 值，就可以在页面上看到对应的标签值，这使得非技术人员也可以方便使用标签。

标签值接口查询功能主要用于标签的在线应用，其他系统可以通过在线调用接口的方式获得标签对应的值。

批量标签支持用于为数据分析和建模提供支持，比如在进行 Logistic 回归建模的时候，

标签可以作为模型的输入。

12.3.1 标签映射表

从图 12-4 可以看到，标签映射表关联着标签系统的前端和后端，它存储了标签系统中除标签值外的所有属性，包含对标签层次结构的全部描述信息。

标签系统的层次结构可以很方便地映射到数据后端中，标签系统数据后端存储了所有标签的值，因此可以使用关系型数据库表或者 Hbase 表进行存储。

对于关系型数据库来说，标签主题映射到数据库的 Schema，标签类型映射为数据库中的表名，标签映射为表中的字段。对于 Hbase，则可将标签主题映射为 Hbase 的表名，标签类型映射为表的列族名，标签映射为列名，表 12-5 对此进行了总结。

表 12-5 标签系统层次结构：与数据后端的映射关系

层次结构	关系型数据库	Hbase
标签主题	Schema	表名
标签类型	表名	列族名 *
标签	列名	列名

\* 注：根据 Hbase 的存储原理，一般不推荐一个 Hbase 表中存在多个列族，因此在实际操作中，并不把标签类型映射为列族名，而是将其映射为表名的一部分。

标签映射表即是存储上述映射关系的一个控制表，根据存储在标签映射表中的信息，程序可以自动定位到对应标签的值。

据此，我们将图 12-2 中的客户主题映射为物理存储，选择 Oracle 作为数据后台。客户主题映射为 Oracle 中的 Schema，在 Oracle 服务器中创建一个 Schema (User)，取名为 Client。

该客户主题下有三个标签类型：基础类、行为类、衍生类。在 Client 下面创建三张表，分别取名为 Tag\_BaseInfo、Tag\_Behavior、Tag\_Derive，对应存储基础类标签、行为类标签和衍生类标签（计算公式）。

接下来要将每个主题下的标签映射为表中的字段，以基础类标签为例，其对应的存储表 Tag\_BaseInfo 应该包含性别 (Clt\_Sex)、出生日期 (Clt\_Bth\_Dte)、手机号 (Clt\_Mob\_Nbr)、邮件地址 (Clt\_Eml\_Adr) 等字段。

确定了上述映射方式，标签映射表的设计就水到渠成，标签映射表应该包含以下信息：标签名称、标签主题 ID、标签类型 ID、标签值定位 ID 等。按照星型模型的方式，图 12-5 展示了标签映射表及附加表的 E-R 关系图。

从图 12-5 中可知，通过标签名称即可关联查询出标签的主题信息、标签的类型信息以及标签值的存储位置。

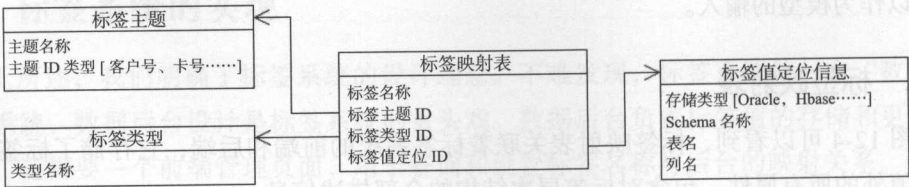


图 12-5 标签映射表的 E-R 关系图

### 12.3.2 标签系统的前端实现

标签系统的前端提供两个功能，分别是标签映射配置和标签值查询，这两个功能点针对两个不同的用户，前者针对标签系统管理员，后者针对标签用户（业务人员或数据工程师、开发工程师等），如图 12-6 所示，虚线为标签管理员参与流程，实线为标签用户参与流程。

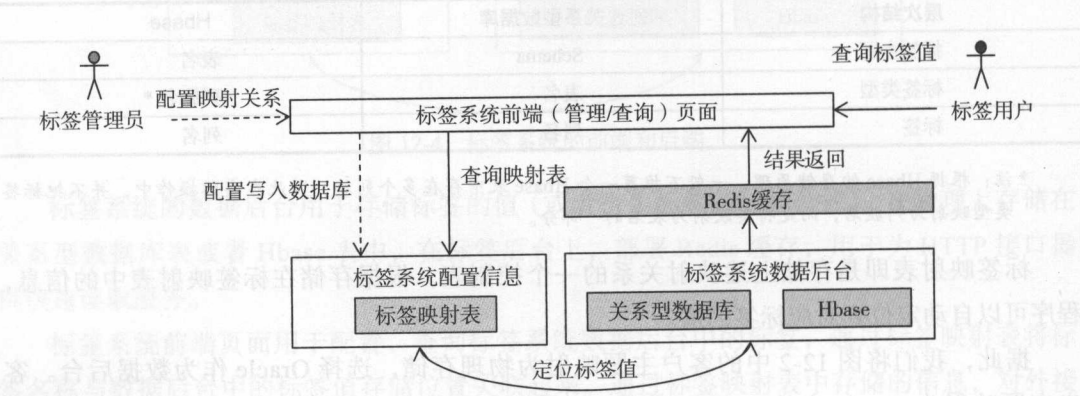


图 12-6 标签系统前端参与者示意图

标签管理员首先通过标签系统前端（管理）页面，配置标签与数据后台的映射关系，这些配置信息存储在标签映射表中。当标签用户在标签系统前端（查询）页面输入查询条件后，系统首先通过标签映射表定位标签的物理位置，然后从标签数据后台读取对应的标签值，并返回页面端并显示出来。

标签系统是一个数据密集型系统，数据后台所占比重很大，前端页面相对较轻，因此对于前端页面，使用 Spring MVC 实现一个简单的 Web 应用即可。由于构建 Web 项目需要 Java 相关的知识，系统的开发可以提交开发工程师实现，本书不详细介绍该 Web 项目的构建过程。

### 12.3.3 标签系统的数据后端实现

标签系统是典型的“数据驱动”的应用系统，数据密集型决定了标签系统数据后端的



设计非常重要，在数据巨大的前提下，仍要保证标签系统的可用性（不间断提供服务）、高效性（快速获得标签值）和可扩展性（方便新增标签）。

如前所述，标签系统的数据后端可能采用两种存储方式：关系型数据库和 Hbase。关系型数据库首选 Oracle 或 DB2，这是因为标签系统数据量很大，使用 Oracle 或 DB2 才可能最大限度地满足系统高效性（MySQL 和 SQL Server 慎用）。下面分别以 Oracle 和 Hbase 为例，设计实现标签系统的数据后端。

### 1. 使用 Oracle

在关系型数据库中，Oracle 以其高性能著称，这也是选择 Oracle 作为标签系统后台的主要原因。仍以图 12-2 中的客户主题为例，要在 Client（客户主题对应的 User）下创建三张表，分别对应客户主题的三个类型：基础类、行为类和衍生类，表名分别为 CLIENT.TAG\_BASEINFO、CLIENT.TAG\_BEHAVIOR 和 CLIENT.TAG\_DERIVE，这些表使用客户主题 ID（客户号）作为主键，并将标签名称作为字段名称。以基础类标签为例，其建表脚本如代码清单 12-1 所示。

代码清单 12-1

```
CREATE TABLE CLIENT.TAG_BASEINFO
```

```
(
```

```
    CLT_ID          NUMBER          NOT NULL          PRIMARY KEY,
```

```
    CLT_SEX         CHAR(1)         NOT NULL,
```

```
    BTH_DTE         DATE            NOT NULL,
```

```
    MOB_PHE         VARCHAR2(11)    NULL,
```

```
    EML_ADR         VARCHAR2(80)    NULL,
```

```
    REG_DTE         DATE            NULL,
```

```
    FST_TRX_DTE     DATE            NULL,
```

```
    .....
)
```

在标签映射表中（请参考图 12-5 标签映射表的 E-R 关系图），需要插入表 12-6 ~ 表 12-9 中所示的内容，以记录标签名称与表 CLIENT.TAG\_BASEINFO 的映射关系。

表 12-6 标签映射表中的内容

标签 ID	标签名称	标签主题 ID	标签类型 ID	标签定位 ID
1	性别	1	1	1001
2	出生日期	1	1	1002
3	手机号	1	1	1003
4	邮件地址	1	1	1004
5	注册日期	1	1	1005
6	首次交易日期	1	1	1006

表 12-7 标签主题表中的内容

标签主题 ID	主题名称	主题 ID 类型
1	客户主题	CLT_ID
2	卡片主题	CRD_NBR

表 12-8 标签类型表中的内容

标签类型 ID	类型名称
1	基础类标签
2	行为类标签
3	衍生类标签

表 12-9 标签定位表中的内容

标签定位 ID	存储类型	Schema 名称	表名	列名
1001	Oracle	CLIENT	TAG_BASEINFO	CLT_SEX
1002	Oracle	CLIENT	TAG_BASEINFO	BTH_DTE
1003	Oracle	CLIENT	TAG_BASEINFO	MOB_PHE
1004	Oracle	CLIENT	TAG_BASEINFO	EML_ADR
1005	Oracle	CLIENT	TAG_BASEINFO	REG_DTE
1006	Oracle	CLIENT	TAG_BASEINFO	FST_TRX_DTE

表 CLIENT.TAG\_BASEINFO 存储真正的标签值，其内容如表 12-10 所示。这样通过标签名称就可以定位到对应的标签值了，例如需要定位客户号 90001 的标签“性别”的值，通过表 12-6 查询到“性别”对应的标签主题 ID、标签定位 ID 分别为 1、1001，然后在表 12-7 中查询到对应的“主题 ID 类型”为“CLT\_ID”，全表名为“CLIENT.TAG\_BASEINFO”，列名为“CLT\_SEX”，程序自动生产并执行 SQL 语句“select CLT\_SEX from CLIENT.TAG\_BASEINFO where CLT\_ID = 90001”，即可得到客户号 90001 对应的标签“性别”的值。

表 12-10 表 CLIENT.TAG\_BASEINFO 的部分内容

CLT_ID	CLT_SEX	BTH_DTE	MOB_PHE	EML_ADR	REG_DTE	FST_TRX_DTE
90001	1	19830919	13721093689	12189@qq.com	20150101	Null
90002	1	19901108	13921093689	chu@hao.com	20160208	20160304
90003	2	19700221	15521093689	Panghu@jdo.com	20090901	20091001

现在已经基本上完成了客户主题下基础类标签的数据后台设计，但是还有一个重要的问题需要考虑，那就是访问效率问题。对于 Oracle 数据库，可以采用表分区的方式提高底层数据的访问效率，下面通过行和列的交叉分区对表 CLIENT.TAG\_BASEINFO 进行改进。

通过对表 CLIENT.TAG\_BASEINFO 的列 CLT\_ID 和列 CLT\_SEX 进行分区，可以缩小数据库查询时的搜索范围，从而提高查询效率。

除了在物理上对 Oracle 进行优化之外，还可以在 Oracle 上使用 Redis 增加一个数据缓冲，可以进一步提高数据的查询效率。

## 2. 使用 Hbase

使用关系型数据库，标签作为表的字段存在，如果一个标签主题下有 1000 个标签，那么需要该表有 1000 个字段，这是一个非常大的“宽表”，其扩展性较差（增加一个标签就需要修改表结构）。

使用 Hbase 可以非常轻松地解决扩展性问题，因为 Hbase 是列存储方式，因此不存在列扩展受限的问题。下面仍以客户主题下的基础类标签为例，讨论基于 Hbase 的存储方式。

根据第 12.3.1 节，我们将客户主题和基础类一起映射为 Hbase 的表名：CLIENT\_BASEINFO，并创建一个列族 CF1，使用主题 ID 作为 RowKey，并将标签 ID 作为列名，Hbase shell 建表命令如代码清单 12-2 所示。

代码清单 12-2

```
hbase(main):001:0> create 'CLIENT_BASEINFO', 'CF1'

CREATE TABLE CLIENT.TAG_BASEINFO
(
  CLT_ID      NUMBER      NOT NULL PRIMARY KEY,
  CLT_SEX     CHAR(1)      NOT NULL,
  .....
  PARTITION BY RANGE(CLT_ID) SUBPARTITION BY LIST (CLT_SEX) (
    PARTITION P1 VALUES LESS THAN(100000) TABLESPACE tagbaseinfo (
      SUBPARTITION P1SUB1 VALUES (1) TABLESPACE tagbaseinfo,
      SUBPARTITION P1SUB2 VALUES (2) TABLESPACE tagbaseinfo
    ),
    PARTITION P2 VALUES LESS THAN (200000) TABLESPACE tagbaseinfo (
      SUBPARTITION P2SUB1 VALUES (1) TABLESPACE tagbaseinfo,
      SUBPARTITION P2SUB2 VALUES (2) TABLESPACE tagbaseinfo
    )
  )
```

这种情况下，标签定位表中的内容如表 12-11 所示，其中列名对应表 12-6 中的标签 ID。此时，如需定位客户号 90001 的标签“性别”的值，通过表 12-6 查询到“性别”对应的标签主题 ID、标签定位 ID 分别为 1、1001，然后在表 12-7 中查询到对应的“主题 ID 类型”为“CLT\_ID”，在表 12-11 中查询到表名为“CLIENT\_BASEINFO”、列名为“1”，程序通过执行命令“get ‘CLIENT\_BASEINFO’ , ‘90001’ , ‘CF1:1’”，即可得到客户号 90001 对应的标签“性别”的值，其中 CF1:1 表示列族 CF1 下面的列“1”（对应标签 ID）。

至此，Hbase 作为标签的数据后台设计已完成，显然它比使用 Oracle 要简单易行，且理论上具有无限扩展性，我们推荐将 Hbase 作为标签系统的数据后台。

表 12-11 标签定位表中的内容

标签定位 ID	存储类型	Schema 名称	表名	列名
1001	Hbase	N/A	CLIENT_BASEINFO	1
1002	Hbase	N/A	CLIENT_BASEINFO	2
1003	Hbase	N/A	CLIENT_BASEINFO	3
1004	Hbase	N/A	CLIENT_BASEINFO	4
1005	Hbase	N/A	CLIENT_BASEINFO	5
1006	Hbase	N/A	CLIENT_BASEINFO	6

同样，为了进一步提升查询性能，可以在 Hbase 上增加 Redis 作为数据缓存，具体方法可参考相关书籍。

### 12.3.4 标签系统的在线接口实现

标签系统的在线接口是为其他系统调用的，是标签在线应用的基础，可以通过 SpringMVC 框架搭建一个 Web 服务即可实现该功能。为简单起见，可以将接口 API 的请求 URL 按照 REST 规范进行定义，如下：

http://<server>/ 标签主题 ID/ 标签类型 ID/ 标签 ID

这时，如果需查询客户号 90001 的标签“性别”的值，则只要发现如下请求即可获得标签值，结果以 JSON 的格式返回。

GET: http://<server>/1/1/1

在线接口的 Web 实现要具备 Web 应用开发能力，可以向开发工程师寻求帮助，本书不再详述具体的编码过程。

## 12.4 本章小结

标签系统是一个数据密集型的用系统，属于典型的数据驱动的应用，它能够为众多的应用提供支持，如数据建模、数据营销、个性化推荐等。

由于标签系统是相对“底层”的数据应用，系统的运营过程总是伴随大批量数据的迁移和计算，因此标签系统的重点在于标签架构的设计和维护。本章介绍的标签系统是基于银行信用卡系统的实际应用场景而来，有很高的实用价值。

从设计理念上，本章从逻辑上定义了标签系统的层次结构和更新规则。层次结构的设计，主要目的是为了与数据平台之间的映射和系统的自动实现，而更新规则是为了方便标签后台数据系统的数据更新。



标签系统的实现部分，重点是标签的数据后台，本章分别介绍了关系型数据库和 Hbase 数据库的后台实现方式。标签系统的接口设计是为其他系统调用提供支持，使用 Web 接口符合面向服务的系统框架要求，便于跨系统的数据交互。

下一章将介绍一个新的数据应用系统：数据智能营销平台，该平台是完全基于标签系统的数据应用。

章 13 数据智能营销平台

#### 4) 统一管理，便于效果追踪

上述几点是我们设计数据自助营销平台的主要考虑因素，也是业界同类系统开发数据自助营销平台的重要切入点，因此有必要对上述几个方面做进一步深入探讨。

### 13.1.1 自动化营销，提升工作效率

在很多公司，每个业务部门都可能有一些运营人员，他们的职责是通过手工提交工单的形式，向技术部门申请提取数据进行营销。众多的业务运营人员提出五花八门的数据需求，希望数据工程师能够“急速”响应，以便错过转瞬即逝的营销窗口。而现实是数据工程师有限的人力，几乎无法“及时”完成其中任何一项数据需求。以某大型国有商业银行信用卡中心为例，数据部门往往要对接整个公司所有的运营人员，而每个业务部门的运营人员可能同时提交多份数据营销需求，要求数据工程师从数据平台中筛选出符合条件的客户，提供这些客户的手机号、Email 等信息，用于在通知平台上发送短信以进行营销驱动。按照这种模式，整个过程需要至少两周时间才能完成，就像图 13-1 所示的那样。

图 13-1 展示了人工营销流程，以某商业银行信用卡中心为例，运营人员提交营销需求，数据部门进行审核，审核通过后，数据部门会提供客户名单，运营人员再根据名单进行营销。

图 13-1 人工营销流程（以某商业银行信用卡中心为例）

图 13-1 展示了人工营销流程，以某商业银行信用卡中心为例，运营人员提交营销需求，数据部门进行审核，审核通过后，数据部门会提供客户名单，运营人员再根据名单进行营销。

图 13-1 展示了人工营销流程，以某商业银行信用卡中心为例，运营人员提交营销需求，数据部门进行审核，审核通过后，数据部门会提供客户名单，运营人员再根据名单进行营销。

图 13-1 人工营销流程（以某商业银行信用卡中心为例）

## 数据自助营销平台

同样，为了进一步提升查询性能，可以在 HBase 上增加 Redis 作为数据缓存，具体方法可参考相关书籍。

## 12.3.4 标签系统的在线接口实现

标签系统的在线接口是为其他系统调用的，是标签在线应用的基础，可以通过 SpringMVC 框架搭建一个 Web 服务即可实现该功能。为简单起见，可以将接口 API 的请求 URL 按照 REST 规范进行定义，如下：

凡事皆须务本，国以人为本，人以衣食为本。

——吴兢《贞观政要》

没有商品这样的东西。顾客真正购买的不是商品，而是解决问题的办法。

——特德·莱维特

作为数据人员，总是会被问及一个现实的问题：数据如何产生价值？很多情况下，数据通过间接的方式产生价值，而公司决策层却希望看到数据产生的直接价值，这也是很多公司对数据建设缺乏热情的重要原因。

数据产生价值的最直接途径非“数据营销”莫属，数据营销的理念已经提出了很多年，几乎每个公司的领导层都曾可能在会议上提及“数据营销”的概念，但要真正实现数据营销，并不像想象中的那么轻松。

实际上，“数据营销”已经可以算是数据应用的高级形式了，首先必须有一个可靠且高效的数据基础平台，其次要有具备成熟经验的数据挖掘和数据建模团队，最后还要有系统开发应用团队。

当公司具备充分条件时，再来看数据营销，将是一件水到渠成的事情。当然，数据营销同样应该系统化、自动化，这是数据应用系统的又一个实证。

## 13.1 数据自助营销平台的价值所在

一个成熟的数据自助营销平台，具备以下功能。

- 1) 自动化营销，提升工作效率。
- 2) 降低营销成本，提升用户体验。
- 3) 个性化营销，提升响应率。
- 4) 统一管理，便于效果追踪。

上述几点是我们设计数据自助营销平台的出发点，也是说服公司决策层同意开发数据自助营销平台的重要切入点，因此有必要对上述几个方面做进一步深入认识。

### 13.1.1 自动化营销，提升工作效率

在很多公司，每个业务部门都可能有一些运营人员，他们的职责是通过手工提交工单的形式，向技术部门申请提取数据进行营销。众多的业务运营人员提出五花八门的数据需求，希望数据工程师能够“急速”响应，以免错过转瞬即逝的营销窗口。而现实是数据工程师有限的人力，几乎无法“及时”完成其中任何一种数据需求。

目前国内大部分商业银行及信用卡中心，数据部门往往要对接整个公司所有的运营人员，而每个业务部门的运营人员可能同时提交多份数据营销需求，要求数据工程师从数据平台中筛选出满足条件的客户，提供这些客户的手机号、Email 等信息，用于在通知平台上发送信息以进行营销促动。按照这种模式，整个过程需要至少两周时间才能完成，就像图 13-1 所示的那样。

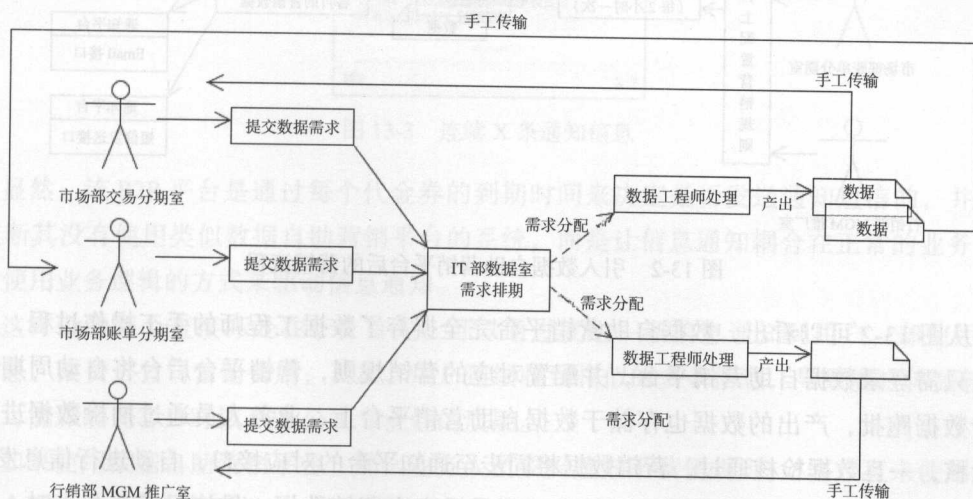


图 13-1 人工营销流程（以某商业银行信用卡中心为例）

在图 13-1 所示的流程中, IT 部数据室是一个效率瓶颈, 由于仅有少量的数据工程师处理大量的数据需求, 因此不得已采取需求排期的方式, 这必然导致大多数数据需求不能及时处理。

流程中的另外一个问题在于, 数据工程师产出数据后, 需通过手工的方式传输相关数据至运营人员, 然后再由运营人员在相应的通知平台上进行数据发送(营销短信、Email 或 App 推送)。其中有大量的手工作业导致了非常高的操作风险, 事实上, 曾经发生过将睡眠户促动信息发送给活跃新户的情况。

从需求提交到最后获得加工后的数据, 业务部门的运营人员不得不等待 2~3 周的时间, 并且一旦出现数据返工的情况, 就会消耗额外的时间, 很可能导致错失营销窗口的情況。例如, 为了进行双十一信用卡促动, 行销部在 10 月 26 日即提交了数据提取需求, 在 11 月 9 日得到了反馈, 但数据检核时发现有部分客户不应该出现在营销列表中, 不得不联系数据工程师进行数据返工, 数据工程师在处理正常需求的同时只好加班加点处理返工的需求。

数据自助营销平台用于上述场景后, 将彻底改变整个过程。图 13-2 展示了引入数据自助营销平台后的新流程。

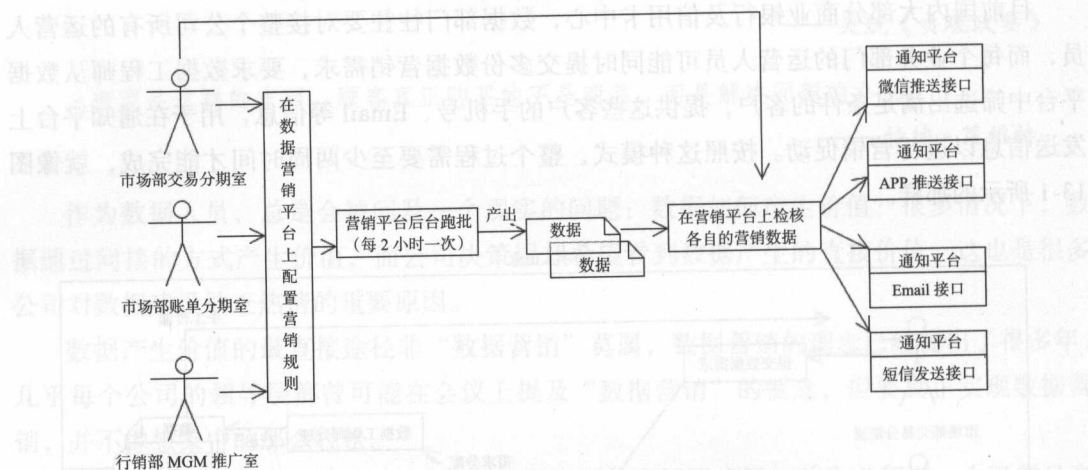


图 13-2 引入数据自助营销平台后的营销流程

从图 13-2 可以看出, 数据自助营销平台完全摒弃了数据工程师的手工操作过程, 业务人员只需登录数据自助营销平台, 并配置对应的营销规则, 营销平台后台将自动周期性地运行数据跑批, 产出的数据也存储于数据自助营销平台上, 业务人员通过抽检数据进行数据检核, 一旦数据检核通过, 营销数据将同步至通知平台的对应接口, 自动进行信息发送。

改进后的流程从业务人员提交需求, 到最后产出营销数据, 最快只需 2 个小时, 从至少 2 周到最快 2 个小时, 其效率有了质的飞跃。整个过程 IT 数据室已经不再是效率瓶颈,



潜在的效率瓶颈在于营销平台后台跑批的机器性能。

通过对接通知平台,同样降低了业务人员手工发送营销信息的操作风险,减少了业务人员的工作量,提升效率的同时节省了业务人员和IT数据人员的人力。

### 13.1.2 降低营销成本,提升用户体验

我们已经看到,数据自助营销平台能够提升工作效率,那么它又如何能够降低营销成本,并提升用户体验呢?这看似矛盾,实际上可以通过数据自助营销平台统一起来。

首先,我们查看笔者曾经收到的某P2P平台发送的通知信息。笔者有7张代金券将要过期,于是在同一天连续收到7条相似内容的代金券过期的短信息,如图13-3所示。



图 13-3 连续 X 条通知信息

显然,该P2P平台是通过每个代金券的到期时间来决定是否发送通知短信的,并且可以推断其没有使用类似数据自助营销平台的系统,而是让信息通知耦合在正常的业务系统中,使用业务逻辑的方式来驱动信息通知。

这样做的两个直接坏处在于:首先,可以通过发送一条信息通知到用户,却使用了7条信息,浪费了公司营销费用;其次,用户连续收到相似的多条信息,导致客户体验较差,甚至可能影响到客户对该P2P公司专业性的质疑。

如果使用数据自助营销平台,那么业务人员要配置如下营销规则:“客户有未使用的有效代金券”且“代金券到期日-当前日期=3的优惠券个数>0”,将营销频率设置为“每天”,并在通知平台配置对应的短信模板为“尊敬的客户,您有7张未使用的代金券即将在3天

后过期,累计金额为  $\times \times \times$  元,请登录平台查看详情”。这样营销平台会周期性(每天)将符合营销规则的客户提取出来,并自动通过通知平台接口将营销信息发送出去,客户只会收到一条信息,通知其有即将过期的代金券未使用。

假设平均每个客户每天收到的通知短信由 5 条降到 1 条,一条短信费用为 0.05 元,每天需发送 1 万个客户,那么每天可节省的短信费用为 2000 元。显然,通过数据自助营销平台,在节省了营销费用的同时,也减少了对客户的骚扰,提升了客户体验。

### 13.1.3 个性化营销,提升响应率

传统的数据营销容易落入一个陷阱:覆盖尽可能多的客户。这种“无差别式”的营销直接导致极低的响应率、过多的客户骚扰,当然也浪费了营销经费。有了数据自助营销平台,营销人员则可以方便考虑个性化的“差异化营销”,这种差异化营销至少有两个方向可以考虑:

- 1) 在同一个营销活动中,不同客户收到的营销信息内容有差别。

- 2) 针对具体的营销活动,通过合理设置营销规则,对高响应率的客户进行促动。

例如,信用卡运营人员准备进行一次圣诞节促销活动,期望通过促销促动消费频率较低的客户积极用卡,并且尽可能地提高卡均消费金额。在通过营销平台设置营销规则的时候,首先限定“近 6 个月用卡次数  $< 10$ ”,然后根据限定客户的“偏好标签”设置奖励,如喜欢购买化妆品的女性客户,设置从圣诞节开始随后的 3 个月内,每月消费满 1800 元即可在第 3 个月免费获得某个化妆品,其中的 1800 元也是根据该客户的月均卡均消费计算出来的(略高于历史上的月均卡均消费额)。通过设置用户偏好的奖励物品,并将达标条件设置成略高于该客户的历史用卡水平,既可以提高用户的响应率,也可以提高用户的卡均消费额。需要明确的是,所有这些规则都可以通过在营销平台上简单配置完成。

另一个案例是,信用卡希望用户使用分期,以便赚取分期手续费和利息,但是不同的信用卡持卡人对分期的接受程度截然不同,有些人对分期乐此不疲,有些人对分期嗤之以鼻,而有些人对分期态度暧昧。显然,对分期乐此不疲的持卡人,并不需要过多的促动(因为他们会自行关注信用卡的分期活动);对分期嗤之以鼻的持卡人,促动的响应率非常低,对他们进行促动将是资源的浪费(这些持卡人需要的是用卡观念的培养,而不是通过简单的信息促动);对分期态度暧昧的持卡人才是真正需要促动的。

该如何找到对分期态度暧昧的持卡人呢?在上面的标签系统中,我们提到了将模型转化为标签的方法,因此可以通过创建一个客户分期意愿评分模型,并将模型评分转化为“分期意愿评分”标签,数据自助营销平台通过引用“分期意愿评分”标签即可划分不同的持卡人。由于数据自助营销平台是一个“规则配置”系统,因此运营人员通过“实验”的方式寻找到合适的“分期意愿评分”的范围,例如通过若干次的小规模促动,发现“分期意

愿评分 between 35 and 55”的持卡人，对分期活动的参与度比之前明显提升，那么就可以认定“分期意愿评分”在 35 分到 55 分的持卡人属于“对分期态度暧昧的持卡人”。

同样，注意到以上过程仅由运营人员进行规则配置，即完成了一次目标客户群寻找，而整个过程，IT 部门的数据人员可能并不知晓！这真是一件让人开心的事情！

#### 13.1.4 统一管理，便于效果追踪

在使用数据自助营销平台之前，信用卡各个业务部门独自提交各自的数据营销需求给 IT 数据人员，一年通常会多达数千份，这些需求彼此之前会出现“客群交叉”，因此数据工程师不得不在处理每个需求时，排除掉近 1 个月内曾经参与过某项促销活动的客户，但由于需求众多，仅限于排除同类型的活动，因此如果两个看起来毫不相干，并且是来自不同业务部门的数据需求，数据工程师并不会对彼此的客群进行互相剔除，这样的结果是活动效果的互相干扰，并且可能导致同时多个活动骚扰同一个客户。

而在数据自助营销平台上配置的营销活动，会被营销平台自动记录，并可以方便发现活动之间的客群交叉，这极大地方便了业务人员对活动效果的追踪，因为可以通过平台查询到同一时间进行的不同活动，发现彼此之间的相互影响。

数据自助营销平台对营销活动进行统一管理的另一个好处在于，数据分析人员可以通过营销活动记录、发现对当前业务波动原因的一些解释。比如，数据分析人员发现，上个月的开卡率比以往明显提高，因此考虑到可能是营销活动带来的影响，这时可以通过营销平台的活动记录查询，查询出上个月进行的所有营销活动，从而进行分析并寻找原因。

### 13.2 数据自助营销平台的实现原则

基于前面提及的 4 个切入点，可以开始着手设计一个数据自助营销平台。我们会将主要精力放在营销平台的设计上（而不是具体实现上），因为这个平台功能相对复杂，因此真正实现它需要一个项目组协同工作一段时间才能完成。作为读者，可以学习本书对平台的设计理念，然后在实际工作中立项实现这个系统。

#### 13.2.1 数据营销活动的节点

一个系统一般分为前端和后端，前端使用 Web 页面，供用户操作，后端为逻辑计算，为前端展示提供逻辑和数据支持。对于数据自助营销平台，前端使用为业务部门的业务人员，后端的责任人为 IT 部门的数据工程师和系统开发工程师。下面从业务人员的角度入手，对数据营销进行逐步抽象和细化，从而完成平台的框架设计。

从运营人员角度考虑一个数据营销方案，首先是确定营销的客户群对象，从整体客户



中圈出部分客户；其次，考虑对选定的客户进行营销动作，如发送短信（SMS）、电子邮件（Email）或 APP 推送。这两个阶段，我们分别称为 Selection 和 Action，这是一个营销活动最大粒度的抽象，如图 13-4 所示。

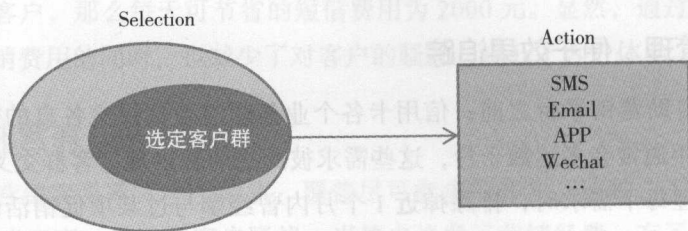


图 13-4 营销活动的两个阶段：Selection 和 Action

进一步对图 13-4 中的 Selection 阶段进行分解，发现选定客户群的方法实际上就是限定客户满足一定的要求。仍以信用卡客户为例，持卡 6 个月以上，并且近 3 个月累计消费笔数达 4 次以上，账单地址在上海……这些都是对客户群的限定。对这些限定进行抽象，可以得到图 13-5 中所示的 Selection 细分模型。

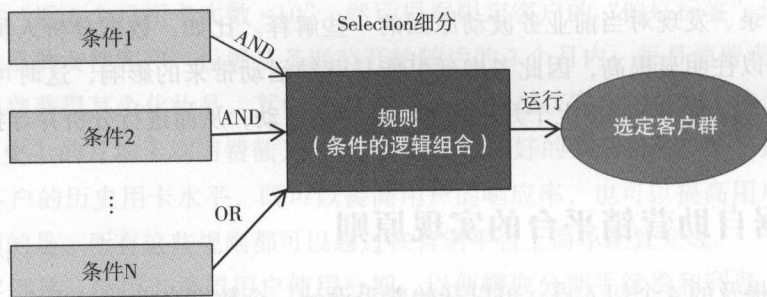


图 13-5 Selection 细分模型

从图 13-5 中可以看到，选定一个客户群，是通过运行一个“规则”来实现的，而“规则”是由多个“条件”通过逻辑组合构成的。如上述例子中：

条件 1：持卡 6 个月以上。

条件 2：近 3 个月累计消费笔数达 4 次以上。

条件 3：账单地址在上海。

这三个条件通过 AND 组合成最后的规则 A：条件 1 AND 条件 2 AND 条件 3。后台批处理进程将根据规则 A 从全量客群中筛选出满足条件的客户群。

相对于 Selection，Action 部分主要完成对选定客户群的“促动”，因为营销活动本身就是将信息送达客户。当前的信息营销一般包括短信（SMS）、电子邮件（Email）、APP 推送、



微信推送等渠道,因此 Action 部分需要选择其中的一个或者多个渠道,并按照对应渠道的接口要求,产出满足格式要求的作业文件,这些作业文件将自动传输至对应的渠道,并被自动发送出去。

通过上述过程,我们认识到一个数据营销活动包括 Selection 与 Action 两部分。Selection 由不同的条件组合成的规则构成,我们将数据自助营销平台的模型整合起来,得到图 13-6 所示的平台逻辑架构,系统设计人员可以根据该架构进行页面逻辑流程设计。

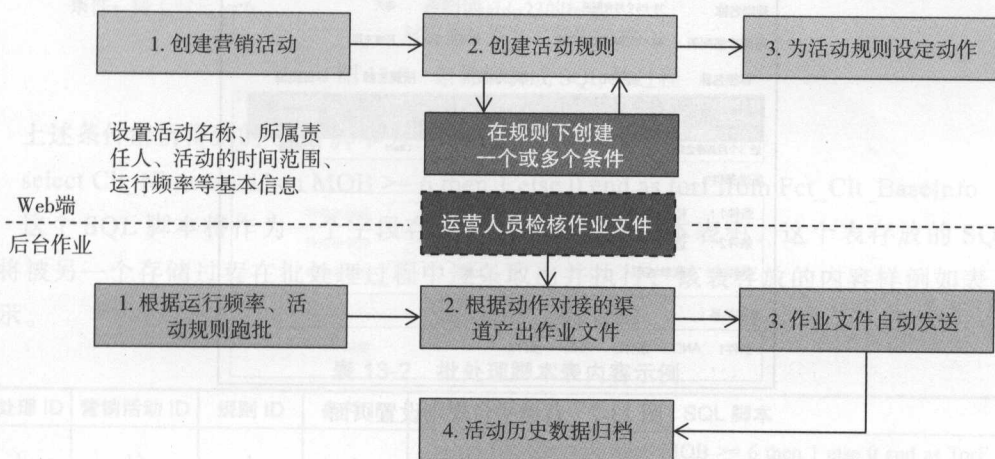


图 13-6 数据自助营销平台逻辑架构

运营人员登录营销平台,首先创建一个空的营销活动,然后在此营销活动下创建一个或多个规则,每个规则下创建不同的条件组合,最后为创建完成的规则设定一个动作。这样就完成了一个基本的数据营销活动配置。

### 13.2.2 数据自助营销平台的基础：标签系统

我们已经将营销活动抽象为 Selection 和 Action 两个步骤,并将其融入图 13-6 的逻辑架构中。要使得营销自助平台能够满足“自助”的要求(即业务人员在页面上进行配置,IT 数据人员尽可能不参与),还有一个重要的环节需要解决:活动规则下的条件从何而来?答案就是:条件来源于标签系统。条件是由标签设置而成的,因此数据营销平台需要与标签系统进行对接,从标签系统中读取所有可供使用的标签信息,并显示在自助营销平台的条件设置页面。仍以上述规则 A 中的三个条件为例,与标签的对应关系如表 13-1 所示。

所以,为了配置该规则,需要表 13-1 中对应的三个客户主题下的标签:MOB、Trx\_Cnt\_R3m、Adr\_Cty。这三个标签分别属于客户主题下的基础标签、行为标签和衍生标签。运营人员在营销平台的规则设置页面选择需要的标签,并设置对应的条件,这些条件之间

默认以 AND 连接, 可以通过手工编辑逻辑关系进行调整, 图 13-7 展示了这个关键页面。

表 13-1 条件与标签对应表

条 件	对应标签名称	转换为标签后的条件
持卡 6 个月以上	持卡时长 (MOB)	MOB >= 6
近 3 个月累计消费笔数达 4 次以上	近 3 个月累计消费笔数 (Trx_Cnt_R3m)	Trx_Cnt_R3m >= 4
账单地址在上海	账单城市 (Adr_Cty)	Adr_Txt in ( “上海” )

图 13-7 营销平台规则设置页面

由此可见, 标签系统需要提供尽可能多的公用标签, 以满足尽可能多的规则 (条件) 设置要求, 一旦业务人员发现条件需要的某个标签不存在, 就需要向 IT 数据人员提交新增标签需求, 当 IT 数据人员将新的标签增加到标签系统之后, 业务人员就可以在营销管理平台的上述页面看到并使用该标签。

页面提交时, 规则对应的逻辑关系保存至后台数据库, 其中的标签以标签 ID 的形式进行保存, 通过标签 ID 可以进一步寻找标签所在的事实表, 并取得对应的标签值。最终的逻辑关系值如果是 true, 则表明该客户满足该规则; 如果结果为 false, 则表明该客户不满足该规则, 因此不属于目标客户群。

下面将进一步探讨规则值 (即逻辑关系值) 的批量计算过程。

### 13.2.3 数据自助营销平台的批量任务

上面将营销活动的规则通过页面转换成了条件之间的逻辑组合, 后台批量处理的任务就是为每个客户计算出逻辑组合的值, 并筛选出逻辑值为 true 的客户作为目标客户。为了便于追踪, 我们先计算出每个条件的布尔值, 并保存在数据库中, 然后再计算出整个条件组合的布尔值。

为了体现出数据库批量计算的优势, 最好的方式是将条件解析成 SQL 语句, 然后执行

该 SQL 语句, 得出该条件的布尔值。以条件“持卡时长  $\geq 6$ ”为例, “持卡时长”是标签平台中的一个标签, 假设其标签 ID (TagId) 为 23, 通过该标签 ID 可以查询出该标签值所在的表名 (Fct\_Clt\_BaseInfo) 和列名 (MOB)。整个解析过程如图 13-8 所示。

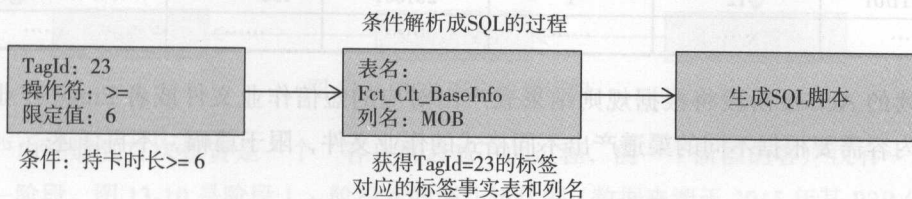


图 13-8 条件解析成 SQL 的过程

上述条件解析得到的 SQL 脚本为:

```
select Clt_Nbr,case when MOB >= 6 then 1 else 0 end as torf from Fct_Clt_BaseInfo
```

这个 SQL 脚本将作为一个字段存放在一个批处理脚本表中, 这个表存放的 SQL 脚本将被另一个存储过程在批处理过程中逐条取出并执行, 该表存放的内容样例如表 13-2 所示。

表 13-2 批处理脚本表内容示例

批处理 ID	营销活动 ID	规则 ID	条件 ID	SQL 脚本
1	12	1	1	select Clt_Nbr,case when MOB >= 6 then 1 else 0 end as TorF from Fct_Clt_BaseInfo
2	12	1	2	select Clt_Nbr,case when Trx_Cnt_R3m >= 4 then 1 else 0 end as TorF from Fct_Clt_BaseInfo
3	12	1	3	select Clt_Nbr,case when Adr_Cty in ( '上海' ) then 1 else 0 end as TorF from Fct_Clt_BaseInfo

这样, 批处理存储过程从表 13-2 中逐条取出 SQL 脚本字段, 运行后将结果存放在另外一个结果表中, 如表 13-3 所示。

表 13-3 批处理条件结果表内容示例

批次日期	批处理 ID	客户号	结果
2015/11/01	1	100001	1
2015/11/01	1	100002	0
		.....	
2015/11/01	2	100001	0
		.....	

从表 13-3 可以发现, 每个客户号对应的每个条件都计算出了结果 (1 或 0, 对应 True 或者 False), 然后根据逻辑关系表即可计算出所有规则结果为 True 的客户号, 这些客户号连同其对应的属性信息存放在规则结果表中, 内容如表 13-4 所示。



表 13-4 规则结果表

批量日期	营销活动 ID	规则 ID	客户号	手机号	Email
2015/11/01	12	1	100034	135*****	quezy@test.com
2015/11/01	12	1	201001	136*****	× × × × × @test.com
.....	.....	.....	.....	.....	.....

后续的 Action 阶段将根据规则结果表产出对应的短信作业文件或者 Email 作业文件，这部分内容需要根据不同的渠道产出不同格式的作业文件，限于篇幅，不再详述。

13.2.4 实时数据营销

本章之前的内容涉及的营销都是批量营销，即系统周期性地 进行批处理，然后批量产出营销作业档，并对客户进行营销。这是一种传统的营销方式，同时可以方便应用于“无差异”场景，如前面提到的抵用券过期短信问题，就可以通过批量营销的方式解决。

实时数据营销也叫“基于场景的营销”，营销内容与客户所处的“场景”紧密结合，例如，当你到加油站加油，刷卡付款后，就会收到某汽车润滑油的促销信息；如果你手机中安装了某 P2P 公司的 APP，靠近该 P2P 公司的门店时，就会收到门店推送的定制化投资理财信息。

实时营销的数据处理方式由原来的批处理改为“实时处理”，即对单个客户应用营销规则，这本质上与批量营销没有区别，但是从技术上来看，需要引入“消息队列”。目前的 Kafka、ActiveMQ 等均可以很好地支持这种营销场景（参考第 1 章图 1-3）。消息队列将用户基于场景的“动作”转换为一个“消息”供数据营销平台“消费”，数据营销平台接收到该消息后，依据之前配置的营销规则实时进行处理，并实时反馈营销结果给客户。

实时化是数据营销平台的发展趋势，在搭建数据营销平台时需要重点考虑实时营销与批量营销之间的框架兼容问题，这往往需要系统架构师的参与，这里不再进一步阐述。

13.3 数据自助营销平台的场景实例

13.3.1 客户生命周期管理

客户生命周期管理是数据营销的一种特殊场景。很多公司将客户生命周期管理单独作为一个系统进行运营，实际上使用本书中的数据营销平台，完全可以涵盖主要的客户生命周期场景。下面以一家 P2P 公司客户生命周期维护为例，来具体剖析数据营销平台如何为客户生命周期维护提供服务。

一个典型的 P2P 客户生命周期包含图 13-9 所示的 6 个阶段。



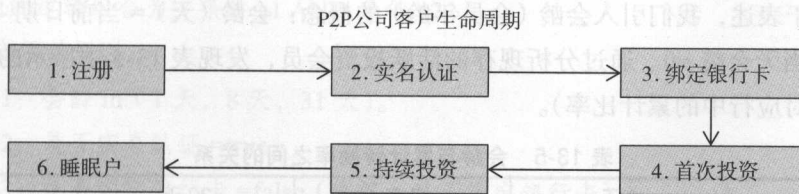


图 13-9 P2P 客户生命周期

客户由注册到首次投资是一个“客户群衰减”的过程，前一个阶段的客户仅有一部分流转到下一阶段。图 13-10 是阶段 1 ~ 阶段 4 的漏斗转化图（数据来源于 2015 年某 P2P 公司）。

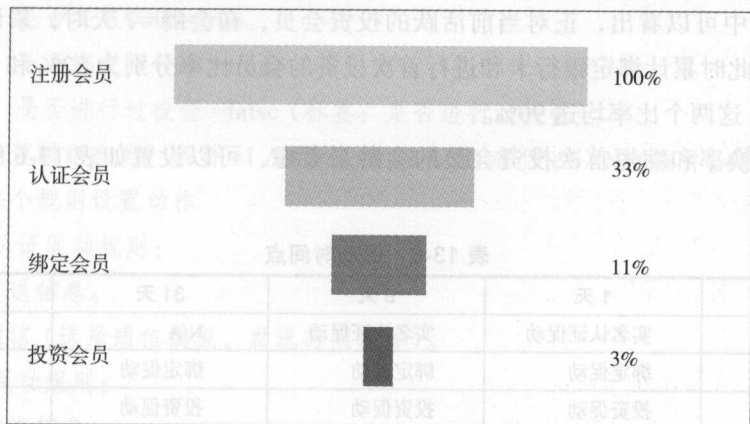


图 13-10 客户转化漏斗图

从图 13-10 可以看出，在所有注册会员中，约有 33% 的会员会进行认证，约 10% 的客户绑定了银行卡，约 3% 的会员进行了投资。每个阶段到下一个阶段的转化率为 30% ~ 33%。

作为公司的运营人员，显然希望注册会员能够尽可能多地转化为投资会员。当客户注册成为会员后，通过发送短信的方式促动注册会员进行认证、绑定和投资。对注册会员进行短信促动，存在图 13-11 所示的促动窗口，现在需要做的是确定每个窗口的截止时间点。

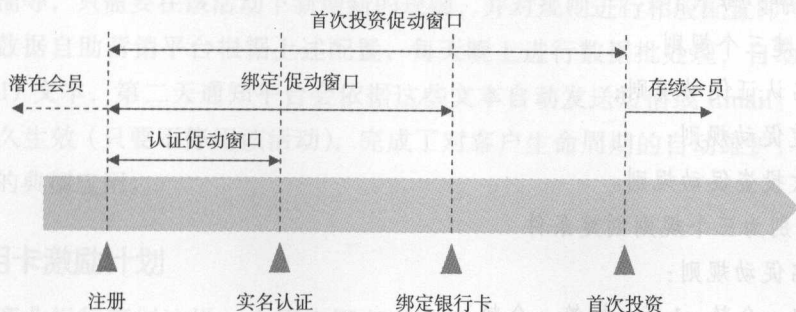


图 13-11 促动窗口示意图

为了便于表述,我们引入会龄(会员年龄)的概念:会龄(天)=当前日期-注册日期。显然,注册当天会龄=0。通过分析现存的活跃投资会员,发现表13-5中所示的数据(表中所示比率为对应行中的累计比率)。

表 13-5 会龄与累计转换率之间的关系

	0 天	7 天	30 天	180 天
实名认证	96%	98%	99%	99.7%
绑定银行卡	67%	82%	90%	97%
首次投资	48%	76%	90%	98%

从表13-5中可以看出,正对当前活跃的投资会员,在会龄=7天时,累计实名认证的比率达98%,此时累计绑定银行卡和进行首次投资的会员比率分别为82%和76%,当会龄达到30天后,这两个比率均达90%。

从提高转换率和缩短首次投资会员的会龄来考虑,可以设置如表13-6所示的促动时间点。

表 13-6 促动时间点

	1 天	8 天	31 天	181 天
实名认证	实名认证促动	实名认证促动	N/A	N/A
绑定银行卡	绑定促动	绑定促动	绑定促动	N/A
首次投资	投资促动	投资促动	投资促动	N/A

注:实名认证促动仅针对注册未实名认证的会员,绑定促动仅针对实名认证后未绑定银行卡的会员,投资促动类推。

根据表13-6,可以在数据自助营销平台上按以下步骤进行配置。

### (1) 创建活动

活动名称:会员生命周期管理;

时间范围:永久;

运行频率:每天。

### (2) 创建三个规则

a) 实名认证促动规则;

b) 绑定促动规则;

c) 首次投资促动规则。

### (3) 分别为三个规则创建条件

a) 实名促动规则:

条件 a1: 会龄=1 天(标签:会龄)。

条件 a2: 是否实名认证=false(标签:是否实名认证)。

条件间的逻辑组合为：条件 a1 AND 条件 a2。

b) 绑定促动规则：

条件 b1：会龄 in (1 天, 8 天, 31 天)。

条件 b2：是否实名认证 =true。

条件 b3：是否绑定银行卡 =false (标签：是否绑定银行卡)。

条件间的逻辑组合为：条件 b1 AND 条件 b2 AND 条件 b3。

c) 首次投资促动规则：

条件 c1：会龄 in (1 天, 8 天, 31 天)。

条件 c2：是否实名认证 =true。

条件 c3：是否绑定银行卡 =true。

条件 c4：是否进行过投资 =false (标签：是否进行过投资)。

条件间的逻辑组合为：条件 c1 AND 条件 c2 AND 条件 c3 AND 条件 c4。

(4) 为每个规则设置动作

a) 实名认证促动规则：

动作：发送信息。

渠道：短信 (选择短信模板、发送时间段等)。

b) 绑定促动规则：

动作：发送信息。

渠道：短信。

c) 首次投资促动规则：

动作：发送信息。

渠道：短信、Email。

经过以上步骤，完成基于数据营销平台的客户生命周期管理活动配置。该活动仅关注会员的认证、绑定、首次投资等时间点，如果想加入其他时间点，如睡眠户促动时间点、会员生日祝福等，只需要在该活动下新增新的规则，并对规则进行相应配置即可。

这样，数据自助营销平台根据上述配置，每天晚上进行数据批处理，自动批量产出短信 (或 Email) 文本，第二天通知平台会依据这些文本自动发送短信或 Email。活动只需一次配置，永久生效 (只要不停用该活动)，完成了对客户生命周期的自动维护，这是数据自助营销平台的典型应用。

### 13.3.2 用卡激励计划

假设某商业银行策划这样一个用户用卡激励活动：2015 年 12 月 1 日 ~ 2016 年 1 月 3 日，在上海地区使用 62 开头的银联信用卡刷卡消费，单笔满 188 元，累计满 3 笔送剃须

刀；累计满 6 笔送拉杆箱；累计满 9 笔送双立人刀具。2015 年 10 月~2015 年 11 月期间参与活动 A 的用户，不参与此活动。

另外，该活动还附加一些短信促动，当用户在此期间累计刷卡满一定笔数，将会收到相应的短信告知客户目前的刷卡状态，如表 13-7 所示。

表 13-7 短信促动对照表

笔数	短信内容
2	您已累计 2 笔满 188 元刷卡消费记录，2016 年 1 月 3 日前再进行 1 笔满 188 元刷卡消费，即送你电动剃须刀。刷满 6 笔送拉杆箱，礼品不叠加
3	您已累计 3 笔满 188 元刷卡消费记录，获得电动剃须刀赠送资格，2016 年 1 月 3 日前刷满 6 笔每笔 188 元消费，送电动剃须刀，刷满 6 笔送拉杆箱，礼品不叠加。请在 1 月 10 日前登录 ×××× 选择礼品，过期视为放弃赠送资格
5	您已累计 5 笔满 188 元刷卡消费记录，2016 年 1 月 3 日前再进行 1 笔满 188 元刷卡消费，送电动剃须刀。刷满 6 笔送拉杆箱，礼品不叠加。请在 1 月 10 日前登录 ×××× 选择礼品，过期视为放弃赠送资格
6	您已累计 6 笔满 188 元刷卡消费记录，获得拉杆箱赠送资格，2016 年 1 月 3 日前刷满 9 笔送双立人刀具，礼品不叠加。请在 1 月 10 日前登录 ×××× 选择礼品，过期视为放弃赠送资格
8	您已累计 8 笔满 188 元刷卡消费记录，2016 年 1 月 3 日前再进行 1 笔满 188 元刷卡消费，即送双立人刀具。请在 1 月 10 日前登录 ×××× 选择礼品，过期视为放弃赠送资格
9	您已累计 9 笔满 188 元刷卡消费记录，获得双立人刀具赠送资格，请在 1 月 10 日前登录 ×××× 选择礼品，过期视为放弃赠送资格。刷银联卡，生活更精彩

短信发送部分实际上是一个实时营销的场景，因为最好的情况是当客户刷卡后进行即时提醒，不过我们首先使用批量营销的方式通过数据自助营销平台完成上述工作，随后再来讨论一下实时营销的方式。

在数据自助营销平台上按以下步骤进行配置。

#### (1) 创建活动

活动名称：2015 年末用卡激励计划。

活动时间范围：2015 年 12 月 1 日~2016 年 1 月 3 日。

运行频率：每天。

#### (2) 在“2015 年末用卡激励计划”活动下创建规则

a) 刷卡 2 笔。

b) 刷卡 3 笔。

c) 刷卡 5 笔。

d) 刷卡 6 笔。

e) 刷卡 8 笔。

f) 刷卡 9 笔。

#### (3) 为每个规则创建条件

公共条件 1：参与过的活动=A（标签：参与过的活动）。



公共条件 2: 活动的时间范围: 2015 年 10 月 1 日 ~ 2015 年 11 月 30 日 (标签: 活动的时间范围)。

公共条件 3: 每笔金额  $\geq 188$  元 (标签: 每笔金额)。

此处使用了“参与过的活动”这个标签, 因此需要在标签后台提前准备好一个“近 12 个月活动参与情况标签表”, 用于记录一年内客户参与过的所有活动, 以及活动的相关信息。

a) 规则名称: 刷卡 2 笔。

条件 a1: 累计刷卡笔数 = 2 (标签: 累计刷卡笔数)。

条件间的逻辑组合为: NOT (公共条件 1 AND 公共条件 2) AND 公共条件 3 AND 条件 a1。

b) 规则名称: 刷卡 3 笔。

条件 b1: 累计刷卡笔数 = 3。

条件间的逻辑组合为: NOT (公共条件 1 AND 公共条件 2) AND 公共条件 3 AND 条件 b1。

其余规则类同。

(4) 为每个规则创建动作

a) 规则名称: 刷卡 2 笔。

动作: 发送信息。

渠道: 短信 (选择短信模板 1、发送时间段: 10:00~11:30)。

b) 规则名称: 刷卡 3 笔。

动作: 发送信息。

渠道: 短信 (选择短信模板 2、发送时间段: 10:00~11:30)。

其余动作设置与此类同。

经过上述过程, 我们已经完成了一个相对复杂的数据自动营销, 一次设置, 自动执行。如果使用传统的人工跑数据方式, 那么在 2015 年 12 月 1 日 ~ 2016 年 1 月 3 日期间, 需要在每天的 10:30 前通过人工运行 SQL 脚本的方式准备好所有的短信数据, 期间还有若干个周末, 对于数据工程师来说, 这真是一个无聊而郁闷的工程! 但是通过使用数据自助营销平台, 解决了数据工程师的痛苦, 同时也保证了数据的质量 (排除了人工操作风险)。

尽管使用上述批量处理的方式解决了用卡激励计划, 但最好的方式是使用实时数据营销的方式。因为使用批量营销的方式, 需要 T+1 (即第二天) 触达客户, 造成信息延迟, 最好的方式是在客户刷卡后, 可以即时获得短信促动, 即“基于场景”的信息促动。

如果使用这种“基于场景”的促动, 需要将用户的刷卡交易行为从主机系统 (或者信用卡授权系统) 抛出至消息队列 (如 Kafka) 中, 然后数据自助营销平台作为 Kafka 消息的消费者, 对接收到的交易“消息”进行处理, 来实时进行短信促动。其基本流程与批量处理基本一致, 只不过数据的来源是消息队列中的“实时”数据。

## 13.4 本章小结

数据自助营销平台是典型的数据驱动的应用系统，该系统要做的不仅仅是基于数据的营销，而且还是自动化（去人工）地进行营销，这可以让企业在节省营销人力的前提下同时提高营销效率。

构建一个数据自助营销平台，基础点在于分析并理清各个营销活动的流程和共性，从中提炼出可以“共用”的部分，组成模块让后续流程直接使用，比如一些公共的统计指标，就可以交给预先搭建的标签系统来提供。

数据自助营销平台除了提供基础的计算指标外，还有非常重要的一点是可以进行灵活且简洁的配置，即可以满足大部分营销场景的要求，让系统使用者的使用成本降到最低，这就要求系统的前端交互符合营销人员的“口味”，因此在分析营销活动之外，需要了解公司的营销人员，倾听他们的诉求，从而在产品设计上进行考虑。这也是数据产品经理需要做的事情。

数据自助营销平台可以应用于各个方面，其中个性化推荐系统也可以看作数据营销的一个分支。下一章将介绍基于 Mahout 的个性化推荐系统，并通过实际数据进行规模和效率上的讨论。

## 基于 Mahout 的个性化推荐系统

彼节者有间，而刀刃者无厚；以无厚入有间，恢恢乎其于游刃必有余地矣。

——庄子《庖丁解牛》

营销学不仅适用于产品与服务，也适用于组织与人，所有的组织不管是否进行货币交易，事实上都需要搞营销。

——菲利普·科特勒

个性化推荐是一个典型的数据驱动的应用，它基于已有的用户行为数据，将物品与用户关联起来并呈现给用户。个性化推荐的数据密集特征，使其在互联网领域中获得了诸多应用。现在，几乎所有的电子商务网站都使用个性化推荐引擎，为用户推荐可能感兴趣的物品；信息服务网站也在大量使用推荐引擎，从大量信息中为用户推荐感兴趣的信息；而在数据营销中，个性化精准营销也离不开个性化推荐的参与。

个性化推荐的理念已经提出了很多年，并且已经发展出很多成熟的推荐算法。在实际应用中，算法本身并不是问题，问题在于算法的实现和系统化应用上。过去，你可能需要自己编码从底层实现一个完整的推荐算法，并且往往还面临推荐效率的问题。现在 Mahout 提供了诸多的推荐引擎算法，并且根据推荐算法本身对数据结构进行了优化，因此在大数据量场景下表现优异，非常适合实时推荐引擎的系统化应用。

学习本章，你不需要掌握那些“高深”的数学知识，甚至不需要知晓推荐算法的具体细节，因为这些已经被 Mahout 封装在各个 java 类中，你所要具备的是对应用场景的理解及与 Mahout 推荐引擎相关的知识。

Mahout 的相关内容可阅读《Mahout 实战》(Mahout in Action) 一书。

## 14.1 Mahout 的推荐引擎

### 14.1.1 Mahout 的安裝配置

在使用 Mahout 的推荐引擎前, 需要对 Mahout 进行简单的安装和配置。Mahout 官网上提供了 Mahout 的源码和编译后的版本, 如果下载 Mahout 源码, 则需要使用 Maven 自己将源码编译成二进制代码。简单起见, 我们下载 Mahout 的二进制版本, apache-mahout-distribution-0.11.0.tar.gz 用于 Linux 系统, apache-mahout-distribution-0.11.0.zip 用于 Windows 系统。

下载后将文件解压缩, 可以看到文件夹中包含一系列 jar 包, 这些 jar 可以导入自己的 Java 工程中, 直接使用 Mahout 提供的各种推荐引擎。这种方式即将 Mahout 用于项目开发的类库, 在 14.2.1 节测试 Mahout 推荐算法的效率时, 使用的即是这种方式。

如果要将 Mahout 部署到 Hadoop 分布式环境, 则需要进行简单配置, 具体步骤如下。

- 1) 在 Hadoop 集群的 Master 节点中解压 apache-mahout-distribution-0.11.0.tar.gz。
- 2) 编辑 ~/.bash\_profile 文件, 在其中加入代码清单 14-1 所示的配置信息(假设 Java 环境已经正确配置)。

代码清单 14-1

---

```
export HADOOP_CONF_DIR=/etc/hadoop/conf
export MAHOUT_HOME=/path/to/mahout
export MAHOUT_LOCAL=
```

---

其中, HADOOP\_CONF\_DIR 用于指定 Hadoop 配置文件所在目录, 以便 Mahout 运行 MapReduce 作业; MAHOUT\_HOME 用于指定 Mahout 所在目录, 即在步骤 1) 中解压后得到的文件路径。文中所示路径为作者测试环境路径, 实际操作中需根据实际情况进行正确的路径配置。

需要注意的一点是, 如果需要 Mahout 运行在分布式环境, 则需要将 MAHOUT\_LOCAL 设置为空, 否则 Mahout 将运行在单机环境(即虽然在 Hadoop 集群上, 但仍只在当前机器运行作业)。实测发现, 在 MAHOUT\_LOCAL=true 或者 MAHOUT\_LOCAL=false 时, Mahout 均运行在单机环境, 只有配置 MAHOUT\_LOCAL=, Mahout 才能正确运行在分布式环境。

3) 经过上述步骤, 就可以检验 Mahout 是否安装配置成功了。检验方法是, 在安装 Mahout 的 Linux 系统 shell 环境中, 键入 mahout, 然后回车, 如果屏幕上打印出代码清单



14-2 所示的内容 (仅截取部分内容), 那么说明 Mahout 已经成功配置为分布式运行模式。

代码清单 14-2

---

```
MAHOUT_LOCAL is not set; adding HADOOP_CONF_DIR to classpath.
Running on hadoop, using /opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/
hadoop/bin/hadoop and HADOOP_CONF_DIR=/etc/hadoop/conf
MAHOUT-JOB: /opt/cloudera/parcels/CDH-5.4.4-1.cdh5.4.4.p0.4/lib/mahout/mahout-
examples-0.9-cdh5.4.4-job.jar
An example program must be given as the first argument.
Valid program names are:
.....
```

---

Running on hadoop 指明了使用的 hadoop 版本, 从中可以看到笔者使用的是 Cloudera 的 CDH5.4 版本, 该版本提供了对 Mahout-Job 的支持。虽然 MAHOUT-JOB 显示该版本是基于 Mahout 0.9 的, 但 Mahout 0.11 仍然能够在其上正常运行。

如果打印的内容如代码清单 14-3 所示, 则说明 Mahout 是单机模式, 需要按步骤 2) 修改 MAHOUT\_LOCAL 配置项。

代码清单 14-3

---

```
MAHOUT_LOCAL is set, so we don't add HADOOP_CONF_DIR to classpath.
MAHOUT_LOCAL is set, running locally
An example program must be given as the first argument.
Valid program names are:
.....
```

---

至此, 已经完成了 Mahout 的安装配置工作, 下面可以开始进一步了解 Mahout 的使用方式了。

### 14.1.2 Mahout 的使用方式

Mahout 提供了推荐算法的分布式实现, 即基于 Hadoop 的 MapReduce 的实现 (目前已经开始转移至基于 Spark), 把很多以前运行于单机上的算法转化为了 MapReduce 模式, 这样大大提升了算法可处理的数据量和处理性能。

尽管如此, Mahout 也提供推荐算法的非分布式实现, 因为在数据量尚未足够“大”的时候, 使用非分布式实现反而可以更好地提供在线服务, 这种情况下, Mahout 一般作为类库使用 (Mahout as library)。而使用分布式实现, 则需要有一个离线计算的过程, 该过程一般使用 Mahout 提供的分布式推荐引擎完成 (Mahout on MapReduce)。图 14-1 展示了这两种方式的区别。

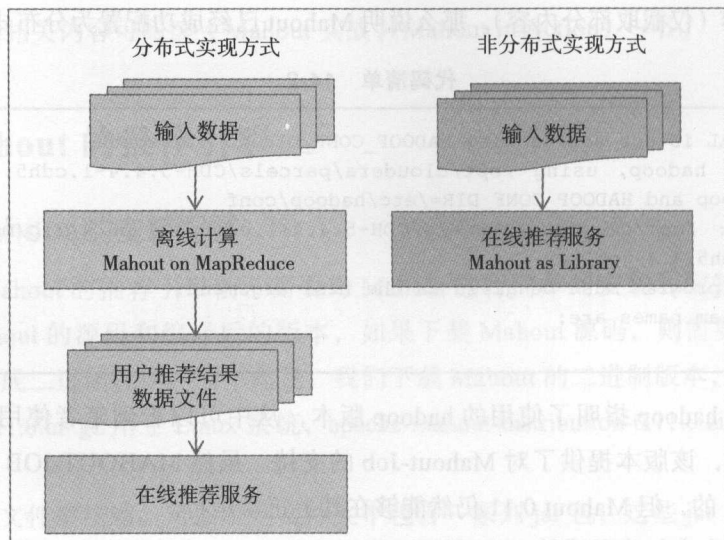


图 14-1 Mahout 推荐引擎的两种使用方式

从图 14-1 可以看到，分布式实现方式需要一个离线计算的过程，在这个过程中，Mahout on MapReduce 作业将读取输入数据文件，计算出所有用户对应的推荐列表，并将这些数据最终存放在推荐结果数据文件中；而在线推荐服务部分才是真正的系统应用，它依据离线计算预先完成的推荐结果文件，快速地返回推荐结果。非分布式实现方式则是通过将输入数据读入在线推荐服务的内存中，然后预先构建出推荐引擎，并通过构建完成的推荐引擎提供推荐服务。

既然存在两种使用方式，那么该如何选择具体的使用方式呢？这往往取决于“规模和效率”。推荐系统必须满足的一个前提是时效性，程序必须相对“实时”地返回推荐结果，一个需要数秒才能返回推荐结果的推荐引擎，已经没有太多应用价值了。

因此，规模和效率是决定使用方式的直接动因。在数据规模较小的时候，效率尚不是问题，因此选择非分布式实现就可以满足要求了。而当数据规模较大的时候，就应该考虑使用分布式实现。14.2 节将使用具体的测试数据来探讨该问题。下面让我们来认识推荐算法中的一个经典算法，了解 Mahout 推荐系统的实现。

### 14.1.3 协同过滤算法

个性化推荐算法中，最著名的算法非“协同过滤算法”莫属。不像传统的内容推荐，协同过滤不需要考虑物品的属性以及用户的行为、行业等问题，只需要建立用户与物品的关联关系即可。该算法实现简单，利于理解，并且对训练数据的要求非常低，实际上，训练数据只需要包含用户 ID、物品 ID、用户对物品的评分即可。Mahout 的个性化推荐引擎

提供了对该算法的全面支持。

协同过滤算法可以基于用户，也可以基于物品。基于用户的协同过滤算法的基本思想是，基于已有的大量用户评分数据，当针对一个用户 A 进行推荐时，首先寻找与 A 相似的用户群 A'，然后通过检查用户群 A' 中每个用户对物品的评分情况，筛选出一部分物品（这些物品应该是用户 A 未曾评价过的）推荐给用户 A。整个过程如图 14-2 所示。

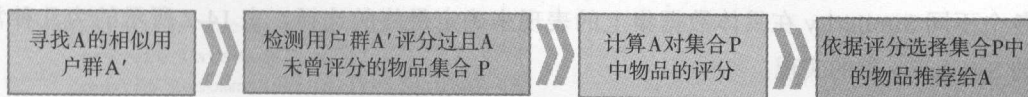


图 14-2 协同过滤算法流程

在图 14-2 中，寻找与 A 相似的用户群，即计算用户 A 与其他用户的相似度，这涉及若干算法，如皮尔逊相关系数法、欧式距离法、余弦距离法、对数似然法、谷本系数法等。不幸的是，这些不同的相似度计算方法会影响到个性化推荐的结果，更加不幸的是，并没有针对不同情况选用对应相似性度量的经验方法，而需要根据具体的训练数据集“挑选”最合适的相似性度量方法。

基于物品的协同过滤算法其流程与基于用户的协同过滤算法类似，只是出发点由用户变为物品。关于协同过滤的更多细节，读者可以很容易从网络上获得相关内容，这对深入理解本章随后的内容会很有帮助。为了节省篇幅，本书对协同过滤算法不做过多介绍。

#### 14.1.4 Mahout 的推荐引擎

Mahout 是一个开源项目，因此其版本处于不断更新中，本书写作时，使用的是 Mahout 的 apache-mahout-distribution-0.11.0 版本。读者可以到 Mahout 的官方网站下载编译好的 binnary 版本或者下载源码，自行编译。

Mahout 实现一个基于用户的推荐，其中涉及四个关键的类（或者接口）：FileDataModel、UserSimilarity、UserNeighborhood 和 Recommender。表 14-1 展示了它们的作用。

表 14-1 推荐引擎关键类和接口的作用

类或接口名称	作用
FileDataModel	在内存中创建训练数据集
UserSimilarity	计算用户相似度
UserNeighborhood	寻找最相似的用户群（邻居）
Recommender	返回推荐结果

如前所述，由于协同过滤算法的特点，其对输入数据的要求非常低，因此 Mahout 对输入数据的要求也非常简单，只需要按照用户 ID、物品 ID、用户对物品的评分这三个信息，即可将数据存放在数据库中，也可以将数据存放在文本文件中（Mahout 建议的方式）。



上述 FileDataModel 类负责读取以文本文件方式存放的数据文件，并将其按照优化后的数据结构存储在内存中。UserSimilarity 和 UserNeighborhood 用于寻找最相似的用户群，Recommender 负责返回最终的推荐结果。

一个推荐系统的关键之处就在于选择合适的相似度（Similarity），针对具体的数据文件，选择不同的 Similarity 类，将影响最终的推荐效果。在实际设计推荐引擎时，一般要通过评估各个不同 Similarity 在具体数据集上的表现来确定最终的选择。表 14-2 所示的为几种常用的 Similarity 类的实现。

表 14-2 几种常用的相似度实现类

类名称	说 明
PearsonCorrelationSimilarity	基于皮尔逊相关系数的相似度
EuclideanDistanceSimilarity	基于欧氏距离的相似度
SpearmanCorrelationSimilarity	基于斯皮尔曼相关系数的相似度
TanimotoCoefficientSimilarity	基于谷本系数的相似度
LogLikelihoodSimilarity	基于对数似然比的相似度

使用 Mahout 创建推荐系统时，上述相似度实现类的选择是一个重要过程，它将直接导致 Recommender 类的表现。可以使用 Mahout 提供的 RecommenderEvaluator 类对 Recommender 进行评分，选择可以使 Recommender 评分达到最优的相似度实现类。具体的评估方式可参考书籍《Mahout in Action》。

图 14-3 是 Mahout 实现基于用户的协同过滤推荐算法的模块流程示意图。图中的 UserSimilarity 是表 14-2 中 Similarity 类的上层接口，它是生成 UserNeighborhood 和 Recommender 的关键输入，这再次印证了 Similarity（相似度）的重要性。

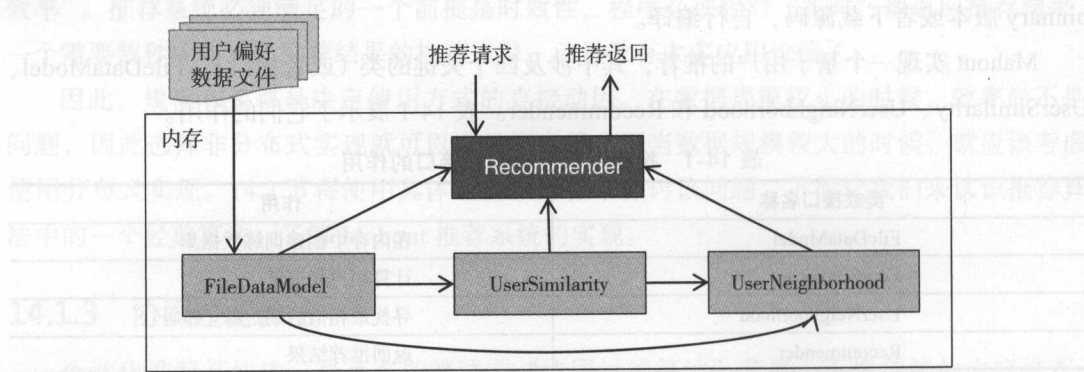


图 14-3 Mahout 实现基于用户的协同过滤推荐算法的模块流程示意图

代码清单 14-4 提供了推荐引擎的一个简单实现（非分布式方式）。在代码清单 14-4 中，new FileDataModel(file) 读入输入数据文件，并在内存中构建 FileDataModel 实例供后续使用；通过 getUserBasedRecommender() 方法最终获得用于推荐的 Recommender 实例。在构



建 Recommender 实例的过程中, 选用 PearsonCorrelationSimilarity 相似度实现类, 并通过 NearestNUserNeighborhood 指定最相近的 N 个用户; getRecommendations() 方法根据传入的 userId 参数和需要推荐给物品个数 itemNum, 并基于之前构建的 Recommender 实例最终返回推荐物品列表 List<RecommendedItem>。

#### 代码清单 14-4

```
public class RecommenderUtil {
    private DataModel model;
    public RecommenderUtil(File file) {
        try {
            this.model = new FileDataModel(file); // 构建 FileDataModel
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    /**
     * 构建 Recommender, 同时使用了 Similarity 实例和 UserNeighborhood 实例
     * @param neighborhoodNum
     * @return GenericUserBasedRecommender
     */
    public Recommender getUserBasedRecommender(Integer neighborhoodNum) {
        UserSimilarity similarity = null;
        UserNeighborhood neighborhood = null;
        DataModel dataModel = this.model;
        try {
            similarity = new PearsonCorrelationSimilarity(dataModel,
                Weighting.WEIGHTED);
            neighborhood = new NearestNUserNeighborhood(neighborhoodNum,
                similarity, dataModel);
        } catch (TasteException e) {
            e.printStackTrace();
        }
        Recommender recommender = new GenericUserBasedRecommender(dataModel,
            neighborhood, similarity);
        return recommender;
    }
    /**
     * 根据 userId、推荐物品个数和 Recommender 返回推荐物品列表
     * @param userId
     * @param itemNum
     * @param recommender
     * @return List<RecommendedItem>
     */
    public List<RecommendedItem> getRecommendations(Long userId, Integer itemNum,
        Recommender recommender) {
        List<RecommendedItem> recommendations = null;
        try {
            recommendations = recommender.recommend(userId, itemNum);
        }
    }
}
```

```
        } catch (TasteException e) {  
            e.printStackTrace();  
        }  
        return recommendations;  
    }  
}
```

以上代码基本是 Mahout 推荐系统非分布式实现的全部代码（不包含评估代码）。从以上代码中可以看到 Mahout 对推荐算法进行了很好的封装，你不需要知晓全部的细节，但仍然可以开发出实用的推荐系统，这正是 Mahout 的优势所在。

## 14.2 规模与效率

现在是时候来面对规模和效率的问题了。前面已经提及，当数据规模较小时，效率不是问题，Mahout 通过非分布式实现即可满足实时推荐的要求。那什么情况下是“规模较小”，什么情况下是“规模较大”呢？

能在网络上找到的对 Mahout 推荐系统的介绍一般都是使用简单的输入，例如一个数十行或者数百行的输入文件，这种情况下无法体会规模导致的效率问题，因为所有的推荐算法都可以在毫秒级别返回推荐结果，这其中还包括读取数据文件的过程。但是在实际场景中，输入文件可能是百万行、千万行甚至数亿行，解决规模导致的效率问题将是推荐系统成败的关键。

### 14.2.1 Mahout 推荐算法的适用范围

Mahout 中基于用户的推荐和基于物品的推荐最常用的实现类分别是 `GenericUserBasedRecommender` 和 `GenericItemBasedRecommender`。这两个实现类的效率如何？

我们通过实际测试进行检验，从非分布式实现方式入手，首先要下载测试数据文件。可以到 <http://grouplens.org/datasets/movielens/> 下载 MovieLens 测试数据集。为了对比，需要下载 MovieLens 1M Dataset、MovieLens 10M Dataset 和 MovieLens 20M Dataset 三个数据集文件，它们分别包含约 100 万条、1000 万条和 2000 万条用户评分数据，解压后用于推荐系统的数据文件（ratings.dat 或 ratings.csv）约 20MB、220MB 和 508MB。其中 MovieLens 10M Dataset 和 MovieLens 20M Dataset 这两个数据集中的 ratings 文件不能直接用于 DataModel 的输入，需要将文件第一行删除，并将文件的列分隔符替换为逗号或 TAB 分割。

数据准备完成后，可以将上述三份数据分别作为输入、训练模型，然后通过调用训练得到的 Recommender 进行推荐，计算为用户进行推荐的平均时间。

当数据文件达到 1000 万个或者 2000 万个时（这取决于运行程序的机器配置），首先遇到的问题可能是 OutOfMemory 错误，这是因为 JVM 自身对程序可使用内存的限制导致的，显然这是数据规模导致的另外一个问题。为了运行测试，通过设置 JVM 参数 -Xmx7168m -Xms5120m，将 JVM 内存的最大使用限制扩充到 7168m（即 7GB，如果机器配置无法提供这么大的内存，也可以将该参数设置为 2048m，但这会使程序消耗更长的时间）。表 14-3 展示了测试结果的对比情况。

表 14-3 基于用户和基于物品推荐的效率对比

	UserBasedRecommender	ItemBasedRecommender
数据量：100 万		
创建 FileDataModel	2296ms	1898ms
创建 ItemBasedRecommender	7ms	25054ms
为用户进行推荐	30ms	1901ms
数据量：1000 万		
创建 FileDataModel	17943ms	16578ms
创建 ItemBasedRecommender	9ms	2ms
为用户进行推荐	108ms	45498ms
数据量：2000 万		
创建 FileDataModel	36526ms	—
创建 ItemBasedRecommender	11ms	—
为用户进行推荐	194ms	—

表 14-3 将每个推荐都分为三个部分：创建 FileDataModel、创建 Recommender 和为用户进行推荐。这是因为创建 FileDataModel 和创建 Recommender 实际上是在程序初始化时完成的，即只需要周期性的初始化（或者当输入数据变化时进行刷新）并将 FileDataModel 和 Recommender 保存在内存中即可，其时间开销并不体现在为用户进行推荐中，因此用户实际感受到的时间是第三部分的时间，它体现了推荐引擎的推荐效率。

对比表 14-3 中的测试数据，显然基于用户的推荐在 100 万数据量时，为用户进行推荐平均耗时 30ms，这在实时推荐系统中是非常不错的表现；当数据达到 1000 万时，其推荐耗时仍能达到 108ms，甚至当达到 2000 万数据量时，仍然保持 194ms 的推荐速度。

但基于物品的推荐，其表现却差强人意。当数据量为 100 万时，其推荐耗时已经接近 2s，这在实际推荐场景中已经接近忍耐极限；当数据达到 1000 万时，推荐耗时达到 45498ms，这已经失去推荐系统的实际作用。尽管 Mahout 官方文档建议使用 GenericItemSimilarity 类，并且预先计算出物品之间的相似度，这种方法虽然提高了推荐效率，但仍然无法达到实时应用的级别。

之所以出现这种明显差异，其部分原因在于基于用户的推荐，其中有 UserNeigh-

borhood 类进行了限制,即不需要所有的相似用户就可以完成推荐,而基于物品的推荐则没有用户领域(或者叫物品领域)的概念,因此需要计算所有相似的物品才能完成推荐。

结合上述测试数据,并考虑到实际情况下机器配置的限制,可以总结出表 14-4 所示的结论。

表 14-4 Mahout 推荐算法在不同规模下的适用情况

输入数据规模	UserBasedRecommender	ItemBasedRecommender
100 万	非分布式 / 分布式	非分布式 / 分布式
1000 万	非分布式 / 分布式	分布式
2000 万	非分布式 / 分布式	分布式
5000 万	分布式	分布式

## 14.2.2 通过分布式解决规模和效率的问题

当面临大规模数据问题时, Mahout 的在线实时推荐面临效率上的挑战,好在 Mahout 本身能够在分布式环境中运行,并且本身已经封装完成部分常用算法的分布式实现。如 item-based recommendations with Hadoop、ALS recommendations with Hadoop。

前面提到, Mahout 的分布式模式的推荐包含离线计算和在线服务两部分。其中离线计算使用 Mahout on Hadoop,解决了数据规模的问题,将大规模的数据计算分布到众多的节点上进行;在线服务使用离线计算的结果,使用 Mahout 提供的 jar 包(Mahout as Library),解决效率问题。

### 1. Mahout on Hadoop

以 item-based recommendations with Hadoop 为例,仍然使用上述 2000 万的输入数据。使用 Mahout 封装后的 recommenditembased,在 Hadoop 集群的 Master 节点(或者 hadoop 客户端)命令提示符下输入如代码清单 14-5 所示的命令。

代码清单 14-5

```
mahout recommenditembased -s SIMILARITY_LOGLIKELIHOOD -i
mahout/input/ratings.csv
-o mahout/output/ml-20 --numRecommendations 25
```

其中, -s 用于指明使用的相似度, -i 用于指明输入文件所在的 hdfs 路径, -o 用于指明 mahout 最终的输出结果路径(hdfs 路径), -- numRecommendations 用于指明每个用户最终返回的推荐物品的最大个数。

上述命令将在 Hadoop 集群上启动若干 MapReduce 服务,运行过程将会花费数分钟至数十分钟(取决于 Hadoop 集群的性能),并在屏幕上打印出非常多的 MapReduce 作业运行



信息，最终在笔者的 Cloudera CDH5.4 环境中，程序打印的最后一行内容如代码清单 14-6 所示。

代码清单 14-6

---

```
15/11/04 11:10:28 INFO driver.MahoutDriver: Program took 1662415 ms
(Minutes: 27.706916666666668)
```

---

从以上代码可以看到，整个过程耗时约 28 分钟，输入 `hadoop fs` 命令查看输出目录，可以看到在指定的输出目录中生成了四个输出文件 `part-r-00000~part-r-00003` 和一个 MapReduce 作业成功标记文件 `_SUCCESS`，如代码清单 14-7 所示。

代码清单 14-7

---

```
[root@hadoop1 ~] #hadoop fs -ls mahout/output/ml-20
Found 5 items
-rw-r--r--  2 root supergroup          0 2015-11-04 11:10
mahout/output/ml-20/_SUCCESS
-rw-r--r--  2 root supergroup    9149610 2015-11-04 11:04
mahout/output/ml-20/part-r-00000
-rw-r--r--  2 root supergroup    9160382 2015-11-04 11:05
mahout/output/ml-20/part-r-00001
-rw-r--r--  2 root supergroup    9149217 2015-11-04 11:05
mahout/output/ml-20/part-r-00002
-rw-r--r--  2 root supergroup    9132622 2015-11-04 11:10
mahout/output/ml-20/part-r-00003
```

---

将输出路径中的输出文件 `part-r-00000~part-r-00003` 从 `hdfs` 导出至 Linux 本地文件系统，通过 `head -3 part-r-00000` 命令查看文件 `part-r-00000` 前 3 行的内容，如代码清单 14-8 所示（为节省篇幅，[] 中的内容未全部列出）。

代码清单 14-8

---

```
[root@hadoop1 ml-20-recommended] # head -3 part-r-00000
4      [481:4.6634183,76077:4.513117,...]
8      [1275:5.0,7235:5.0,...]
12     [4844:4.5094776,8361:4.505701,...]
```

---

从代码清单 14-8 中可以发现，输出的文件每行由两部分组成，中间由“\t”分割，第一部分为用户 ID，第二部分是为该用户 ID 推荐的物品列表和对应的偏好值，如 `481:4.6634183`，表明用户 ID=4 的用户对物品 481 的偏好值为 4.6634183。由于在最初的命令中指定了 `--numRecommendations 25`，因此每个用户 ID 对应最多 25 个物品：偏好值对。

可见，Mahout on Hadoop 基于输入数据，为所有用户预先计算出推荐列表，并对推荐列表中的每个物品给出预测的偏好值。这些推荐结果数据，可供后续在线推荐服务直接使

用，在 Mahout as Library 中介绍这些推荐结果文件的使用方式。

## 2. Mahout as Library

通过离线计算，已经得到所有用户指定数目的推荐物品列表，这些数据存放在文本文件中。作为后续在线推荐服务的输入数据，有多种处理方式，例如可以将文件全部读入内存，或者存放在关系数据库中，当文件特别大的时候，可以考虑将其存放在 Hbase 中，目的是让在线推荐服务高效运行，当输入用户 ID 后，能够毫秒级返回推荐列表。

最为高效的方式，肯定是将文件全部存入内存。如果使用 Java 开发语言，那么你首先想到的可能是 Java 的 HashMap，但 Mahout 有针对性地提供了处理这些数据的专用类：PreferenceArray 和 FastByIDMap，这些类进行了针对性优化，占用更少的内存，因此非常适合大数据量的场景。Mahout 的 0.11 版本中，可以在 mahout-mr-0.11.0.jar 中找到这些类。

将 Mahout 提供的 jar 包导入 Java 项目中，利用 Mahout 封装的类更高效地实现推荐服务，这种方式即 Mahout as Library 的使用方式。代码清单 14-9 和代码清单 14-10 分别提供了将推荐结果文件存储在内存中和 Hbase 中的实现过程。由于上述离线计算所得的四个结果文件的总大小约为 34MB，因此完全可以全部读入内存，以提供快速推荐服务，内存方式是首选方式。

代码清单 14-9

```
public FastByIDMap<PreferenceArray> getItemBasedRecommendationMap(String filepath) {
    File file = new File(filepath);
    FastByIDMap<PreferenceArray> preference = new FastByIDMap<PreferenceArray>();
    if (file.isDirectory()) {
        BufferedReader reader = null;
        String[] filelist = file.list();
        for (String fileStr : filelist) {
            File f = new File(filepath + "/" + fileStr);
            System.out.println("读取文件 " + f.getAbsolutePath() + " 至内存中。");
            try {
                reader = new BufferedReader(new FileReader(f));
                String tempString = null;
                int userIDIndex = 0;
                while ((tempString = reader.readLine()) != null) {
                    String[] strs = tempString.split("\t");
                    long userID = Long.valueOf(strs[0]).longValue();
                    String[] itemPrefs = strs[1].substring(1,
                        strs[1].length()-1).split(",");
                    PreferenceArray prefArr =
                        new GenericUserPreferenceArray(itemPrefs.length);
                    prefArr.setUserID(userIDIndex, userID);
                    for(int i=0;i<itemPrefs.length;i++){
                        String prefSplits[] = itemPrefs[i].
                            split(":");
```

```

        long itemID = Long.
            valueOf(prefSplits[0]).longValue();
        float prefVal = Float.valueOf(prefSplits[1]).
            floatValue();
        prefArr.setItemID(i, itemID);
        prefArr.setValue(i, prefVal);
    }
    preference.put(userID, prefArr);
    userIDIndex ++ ;
}
} catch (IOException e) {
    e.printStackTrace();
}
}
return preference;
}

```

上述代码结合使用了 `FastByIDMap` 和 `PreferenceArray`，首先从文件中读取一行记录，然后将该行数据的第二部分存放在 `PreferenceArray`，并将该行第一部分的用户 ID 和存放有推荐物品列表信息的 `PreferenceArray` 存放在 `FastByIDMap` 中。

生成的 `FastByIDMap<PreferenceArray>` 作为用户进行推荐的基础，常驻内存中。在进行实时推荐时，只需输入用户 ID，即可快速返回对应的 `PreferenceArray`：`PreferenceArray preferenceArray = preferenceArrayMap.get(userID)`。

下面尝试将推荐结果存放在 Hbase 表中，根据推荐结果文件数据的存放方式，可以方便将 Hbase 表的结构设计如下。使用 `UserID` 作为 `RowKey`，包含一个列族 `item`，使用 `ItemID` 作为列名称，`Item` 对应的偏好值作为列的值。

RowKey	ColumnFamily:Column	Value
UserID	ItemID	PreferenceValue

代码清单 14-10 展示了将推荐结果文件生成 `HFile` 的过程，随后将生成的 `HFile` 加载至 Hbase 表中（参阅第 2 章的内容），这里将注意力放在 `Map` 函数生成 `HFile` 的过程上。

代码清单 14-10

```

protected void map(LongWritable key, Text value, Context context) throws
IOException, InterruptedException {
    String line = value.toString();
    String[] items = line.split("\t");
    ImmutableBytesWritable rowkey = new ImmutableBytesWritable(items[0].
        getBytes());
    String[] itemPrefs = items[1].substring(1, items[1].length() - 1).split(",");
    for (String itemPref : itemPrefs) {
        String[] pair = itemPref.split(":");
    }
}

```

```

String column = pair[0];
String prefValue = pair[1];
KeyValue kv = new KeyValue(Bytes.toBytes(items[0]), Bytes.
    toBytes("item"), Bytes.toBytes(column), System.currentTimeMillis(),
    Bytes.toBytes(prefValue));
if (null != kv) {
    context.write(rowkey, kv);
}
}
}

```

Map 函数每次从推荐结果文件中读取一行，并以参数 Text value 的形式传递进来，之后的过程就是从 value 中拆分出 userID、itemID 和偏好值，并将拆分后的数据存放到 org.apache.hadoop.hbase.KeyValue 对象中，然后通过 context.write(rowkey, kv) 将转换后的数据作为 Map 函数的输出写至 HDFS 中。

程序执行结束后，最终存放在表 recommend\_list 中的数据如代码清单 14-11 所示（通过 get 'recommend\_list','4' 查询 UserID=4 的用户对应的推荐物品列表）。

代码清单 14-11

COLUMN	CELL
item:1012	timestamp=1447209415688, value=4.4977155
item:1049	timestamp=1447209415688, value=4.5097203
item:1082	timestamp=1447209415688, value=4.495925
item:1128	timestamp=1447209415688, value=4.50865
item:2051	timestamp=1447209415688, value=4.5111723
item:2373	timestamp=1447209415688, value=4.499443
item:276	timestamp=1447209415688, value=4.493324
item:280	timestamp=1447209415688, value=4.4931173
item:2989	timestamp=1447209415688, value=4.4963245
item:315	timestamp=1447209415688, value=4.4939966
item:33646	timestamp=1447209415688, value=4.499479
item:3688	timestamp=1447209415688, value=4.4995246
item:427	timestamp=1447209415688, value=4.4968843
item:428	timestamp=1447209415688, value=4.495612
item:4681	timestamp=1447209415688, value=4.5056744
item:481	timestamp=1447209415688, value=4.6634183
item:5463	timestamp=1447209415688, value=4.5117846
item:6059	timestamp=1447209415688, value=4.5056353
item:6764	timestamp=1447209415688, value=4.49906
item:7000	timestamp=1447209415688, value=4.499125
item:7340	timestamp=1447209415688, value=4.50107
item:76077	timestamp=1447209415688, value=4.513117
item:77364	timestamp=1447209415688, value=4.5039597
item:782	timestamp=1447209415688, value=4.4974184
item:8633	timestamp=1447209415688, value=4.494288



从以上代码可见，表中的列是以 item：物品 ID 的形式存储的，其中 item 是列族的名称，CELL 中存放的是时间戳（timestamp）和对应的偏好值（value）。在生产环境中，只需要传入 UserID 即可以迅速查询出对应的物品 ID 和偏好值。

## 14.3 实现一个推荐系统

上面已经对 mahout 的使用方式做了探讨，并在此基础上讨论了规模和效率的问题。下面尝试构建一个可以运行在线上生产环境的推荐系统。

### 14.3.1 系统框架

一个完整的推荐系统，一般会包含离线部分和在线部分。离线部分负责数据的收集、清洗，并按照线上部分需要的文件格式生成数据文件。因此其系统框架一般如图 14-4 所示。

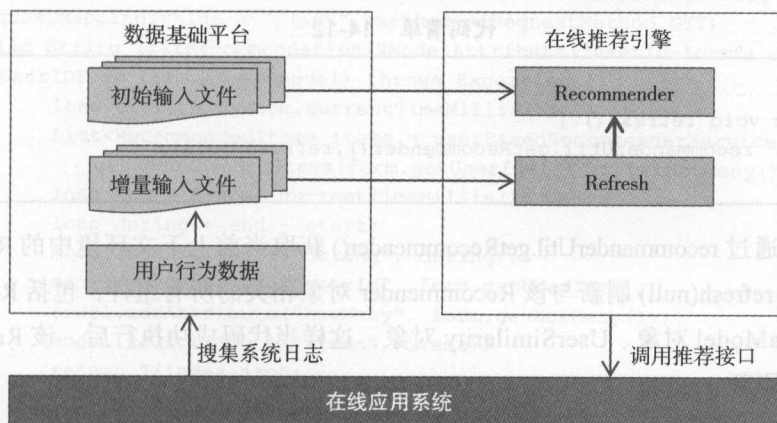


图 14-4 一个推荐系统的框架示例图

离线部分主要通过数据基础平台完成，这个过程是从大量用户行为数据中清理出推荐系统需要的输入数据。一般在初始时，会预先产生一个初始输入文件，然后在线推荐引擎根据初始输入文件训练 Recommender，并将其存放在内存中，对外提供推荐服务。

在线应用系统通过调用推荐接口，即可获得在线推荐引擎的“实时”反馈。同时，在线系统会产生用户行为数据和系统日志等原始数据，这些数据被数据基础平台收集后，经过 ETL 过程产生增量的输入文件，这些增量文件被在线推荐引擎的 Refresh 接口读取，并刷新在线推荐系统的相关模型，从而保证推荐系统的“新鲜度”。

### 14.3.2 推荐系统的刷新

在图 14-4 中已经提到，推荐引擎可以通过 Refresh 接口进行刷新。在这个问题上，Mahout 提供了一种方便的刷新机制，即只要将增量数据文件与原始数据文件以相同的文件前缀命名，并存放在同一个目录下即可，比如初始文件名称为 ratings.dat，那么增量数据文件只要命名为 ratings.1.dat、ratings.2.dat，并同 ratings.dat 放置在同一个目录下，此时通过调用 Recommender.refresh(null) 即可自动将推荐引擎中涉及的一系列对象刷新，包括 DataModel、Similarity、Recommender 等。

需要注意的是，当数据量庞大时，推荐引擎的刷新是一个相对耗时的过程，因此在线刷新推荐引擎需要考虑实际的系统压力问题。实际生产情况下，可以使用定时刷新（如在凌晨业务低峰时进行），或者使用事件触发方式的刷新。无论何种场景，首要保证的是推荐系统的可用性，因此需要选择一个对在线应用影响最小的方式进行。代码清单 14-12 展示了刷新 Recommender 以及相关组件的方法。

代码清单 14-12

---

```
@Override
public void refresh() {
    recommenderUtil.getRecommender().refresh(null);
}
```

---

以上代码通过 recommenderUtil.getRecommender() 获取当前上下文环境中的 Recommender 对象，并通过 refresh(null) 刷新与该 Recommender 对象相关的所有组件，包括 Recommender 对象、FileDataModel 对象、UserSimilarity 对象，这样当代码成功执行后，该 Recommender 引擎就被全部刷新。

### 14.3.3 部署一个可用的推荐系统

现在是时候来部署一个可用的推荐引擎的在线系统了。要让推荐系统运行在在线环境中，需要将其部署在一个 Web 应用框架下，考虑到推荐引擎一般作为后台服务被调用，所以可以将推荐引擎包装为 http 服务，当其他系统发送 http 请求时，推荐引擎服务接收该请求，并返回推荐结果。

为了直观模拟请求和结果返回，以及推荐引擎的刷新，我们实现了一个简单的交互页面，可以通过该页面与后台的推荐引擎进行交互，如图 14-5 所示的页面。

接下来，通过 eclipse 创建一个 Dynamic Web Project，并取名为 CFEngine，为了简单起见，在项目中引入 Spring MVC 和 Hibernate Annotation，你需要把相关的 jar 包全部引入项目中。

**Mahout在线推荐引擎示例页面**

当前使用的数据文件为：E:\Datum\DataFiles\ml-1m\ratings.dat(100万原始数据)，使用UserBased的推荐算法。

你可以将ratings.20m.dat（2000万数据）复制到上述路径，然后点击“刷新引擎”按钮刷新推荐引擎。

在下面输入用户ID和物品个数，点击“提交”按钮进行推荐：

用户ID  推荐物品个数

图 14-5 在线推荐引擎示例页面

由于使用了 Spring MVC 和 Hibernate Annotation 框架，因此只要在 controller 中加入相应的请求地址，并定义处理方法即可，代码清单 14-13 展示了这个过程。

代码清单 14-13

```
@Controller
public class UserBasedRecommenderController {
    @Autowired
    private UserBasedRecommenderService userBasedRecommenderService;
    @RequestMapping(value = "/list", method = RequestMethod.GET)
    public String listRecommendation(@ModelAttribute("userId-form")
        UserIDForm form, Model model) throws Exception {
        long start = System.currentTimeMillis();
        List<RecommendedItem> items = userBasedRecommenderService.
            getRecommendedItems(form.getUserId(), form.getHowMany());
        long end = System.currentTimeMillis();
        long during = end - start;
        model.addAttribute("during", during);
        model.addAttribute("userId", form.getUserId());
        model.addAttribute("howMany", form.getHowMany());
        model.addAttribute("items", items);
        return "/index.jsp";
    }

    @RequestMapping(value = "/refresh", method = RequestMethod.GET)
    public String refreshRecommendation(Model model) throws Exception {
        long start = System.currentTimeMillis();
        userBasedRecommenderService.refresh();
        long end = System.currentTimeMillis();
        long during = end - start;
        model.addAttribute("during", during);
        return "/index.jsp";
    }
}
```

在 Controller 类中通过 @RequestMapping 制定了两个需要处理的请求，分别是 /list 和 /refresh，对应的处理方法分别用于返回推荐列表和刷新推荐引擎。

userBasedRecommenderService 通过 @Autowired 获得一个对象，这个对象在服务启动的时候可进行初始化，而推荐引擎的构建过程正封装在 UserBasedRecommenderService 中，这样 Web 服务在启动的时候，即可进行推荐引擎的初始化，从而保证服务启动后，进行实

时推荐服务。代码清单 14-14 展示了 UserBasedRecommenderService 接口的实现类代码以及 UserBasedRecommenderUtil 的部分代码。

代码清单 14-14

---

```

public class UserBasedRecommenderServiceImpl implements
    UserBasedRecommenderService {
    private UserBasedRecommenderUtil recommenderUtil = new
        UserBasedRecommenderUtil();
    @Override
    public List<RecommendedItem> getRecommendedItems(Long userId, Integer howMany) {
        return recommenderUtil.getRecommendations(userId, howMany);
    }
    @Override
    public void refresh() {
        recommenderUtil.getRecommender().refresh(null);
    }
}

public class UserBasedRecommenderUtil extends RecommenderUtil {
    private FileDataModel model;
    private UserSimilarity similarity;
    private Recommender recommender;
    private static final Log logger = LogFactory.getLog(UserBasedRecommenderUtil.class);
    public UserBasedRecommenderUtil() {
        logger.info(" 初始化 FileDataModel...");
        this.model = super.getFileDataModel();
        logger.info(" 初始化 UserSimilarity...");
        this.similarity = getUserSimilarity(model, "PearsonCorrelationSimilarity");
        logger.info(" 初始化 UserBasedRecommender...");
        this.recommender = getUserBasedRecommender(10);
    }
}
// 其他代码未列出...
}

```

---

由于在服务启动时进行推荐引擎的初始化,因此首先要指定一个默认的输入数据集,为了不使服务启动占用过多时间,我们默认使用 MovieLens 的 100 万条数据集作为初始数据集,并将该数据存放在“E:\Datum\DataFiles\ml-1m\”中。

在 Eclipse 开发环境中,将 Web 项目 CFEngine 部署在 tomcat 环境中,并启动这个 Web 服务,可以在 eclipse 的控制台输出中找到如代码清单 14-15 所示的内容。

代码清单 14-15

---

```

cf.mh.recommender.UserBasedRecommenderUtil.<init>(30) | 初始化 FileDataModel...
org.apache.mahout.cf.taste.impl.model.file.FileDataModel.<init>(185) | Creating
FileDataModel for file
E:\Datum\DataFiles\ml-1m\ratings.dat

```

---



```

org.apache.mahout.cf.taste.impl.model.file.FileDataModel.processFile(357) | Reading file
info...
org.apache.mahout.cf.taste.impl.model.file.FileDataModel.processFile(364) | 
Processed 1000000 lines
org.apache.mahout.cf.taste.impl.model.file.FileDataModel.processFile(368) | Read
lines: 1000209
org.apache.mahout.cf.taste.impl.model.GenericDataModel.<init>(113) | Processed 6040 users
cf.mh.recommender.UserBasedRecommenderUtil.<init>(32) | 初始化 UserSimilarity...
cf.mh.recommender.UserBasedRecommenderUtil.<init>(34) | 初始化 UserBasedRecommender...

```

这些输出信息显示了在服务启动时初始化推荐引擎的过程。当服务启动完成后，在本机打开浏览器，输入 <http://localhost:8080/CFEngine/> 并回车，就可以看到图 14-5 所展示的页面。下面可以尝试输入用户 ID 和需要返回的推荐物品个数进行推荐测试了。

用户 ID 输入 123，推荐物品个数输入 10，单击“提交”按钮，系统会在页面上展示推荐结果，如图 14-6 所示。从图中可以看到，推荐过程耗时 86ms，完全满足实时推荐的时效要求。

用户ID 123	推荐物品个数 10	提交	刷新引擎
为用户123进行推荐，耗时86ms.			
物品ID	偏好值 (预估)		
3949	5.0		
3148	4.5		
3578	4.3210506		
3555	4.3210506		
3948	4.25		
2889	4.0148425		
3301	4.0		
3408	4.0		
3827	4.0		
2628	3.9721072		

图 14-6 推荐引擎为用户 123 推荐的结果

下面尝试刷新这个在线推荐引擎，首先要将 MovieLens 的 2000 万条数据集复制到路径“E:\Datum\DataFiles\ml-1m\”中，并将其重命名为 ratings.20m.dat，现在路径“E:\Datum\DataFiles\ml-1m\”存放有两个文件：原始数据文件 ratings.dat 和新增用于刷新的数据文件 ratings.20m.dat。现在单击“刷新引擎”按钮，耐心等待一段时间后，页面会显示图 14-7 所示的结果。

### Mahout在线推荐引擎示例页面

当前使用的数据文件为：E:\Datum\DataFiles\ml-1m\ratings.dat(100万条原始数据)，使用UserBased的推荐算法。

你可以将ratings.20m.dat (2000万条数据) 复制到上述路径，然后点击“刷新引擎”按钮刷新推荐引擎。

在下面输入用户id和物品个数，点击“提交”按钮进行推荐：

图 14-7 推荐引擎的刷新

推荐系统花费了超过 68s 的时间用于将 2000 条新增数据刷新至推荐引擎，同时这个过程将会消耗大量的内存，因此需要确保机器和 JVM 的可用内存足够。再次输入用户 ID 为

123, 推荐物品个数为 10, 单击“提交”按钮, 图 14-8 展示了推荐结果。

用户ID 123	推荐物品个数 10	提交	刷新引擎
为用户123进行推荐, 耗时434ms.			
物品ID 偏好值 (预估)			
5995	5.0		
7153	4.75		
48394	4.6666665		
79132	4.25		
5618	4.25		
4993	4.25		
58559	4.0		
1588	3.8333333		
1073	3.25		
2004	2.75		

图 14-8 刷新推荐引擎后的推荐结果

与图 14-6 相比, 首个不同在于推荐耗时, 由原来的 86ms 上升为 434ms, 考虑到数据量由 100 万条上升为 2000 万条, 变为原来的 20 倍, 而推荐耗时仅变为原来的约 5 倍, 所以 434ms 仍然是可接受范围。

另外一个不同在于推荐结果, 物品 ID 和预估偏好值均不相同, 从而印证了推荐引擎确实被刷新 (这种大数据量的更新可能导致另一个问题: 该推荐引擎的实现方法在新数据集上已经不是最优的了, 因此你可能需要寻找一种更优的实现算法, 这不在本文的讨论范围之内)。

现在整个推荐引擎已经可以在 Eclipse 开发环境中正常工作了, 你可以自己动手尝试将该 Web 项目导出成 war 包, 并部署在一个服务器的 Tomcat 容器中, 这样就实现了一个完整的在线推荐模拟系统了。

实际的生产环境中, 推荐引擎应该被包装成后台服务, 一般不需要前端展示页面, 并且需要集合一定的业务规则 (比如置顶的物品或者需要排除的物品等), 而推荐算法本身也往往会基于内容进行推荐, 这部分内容可以参考 Mahout in Action 一书中讨论的内容。

## 14.4 本章小结

本章介绍了基于 Mahout 的个性化推荐, 这部分内容在网络上存在非常多的资料, 因此这里仅简单介绍了 Mahout 的安装配置和使用方式, 然后介绍了经典的协同过滤算法。

本章重点讨论了 Mahout 的推荐算法的适用范围, 并进一步讨论了规模和效率问题。在大规模数据的情况下, 就需要通过分布式的方式解决规模带来的效率瓶颈, 由于机器配置限制, 文中的测试结果可能并不能反映企业的实际应用场景, 但文中提供的思路为读者提供了一种参照, 在设计一个推荐系统时, 除了算法本身, 需要重点考虑数据本身, 通过试验找到兼顾效果和效率的实现方案。

下一章将介绍一种基于 Spark Graphx 的图计算引擎, 同时它也是一种新型的推荐系统实现方式。



## 第 15 章

## Chapter 15

# 图计算与社会网络

业精于勤而荒于嬉，行成于思而毁于随。

——韩愈·《进学解》

如果你想有更好的呈现效果，你需要思考设计。

——盖伊·川崎

生活中，我们总在不停地与周边事物发生联系，同时也在一刻不停地产生着数据，这些数据联系着不同的实体，组成一张庞大繁杂的网络。比如，过去一年中，你与同事或者朋友间的聚餐形成一张聚餐关系网，在电商网站上的购物记录形成物品购买网络、银行转账记录和朋友间的通话记录等，所有这些数据形成的网状结构，都可以称为社会网络（social network）。

社会网络是个体之间彼此关联形成的一个网状体系，它是一个网状图。个体称为节点，个体之间的关联称为关系（边），所以社会网络也可以称为关系网络。

过去，关系网络的应用范围受到很大限制。主要原因在于关系网络一般都非常庞大且复杂。传统的关系型数据库在存储多边关系时遇到了困难，同时使用传统的关系数据库范式存储关系网络，则在计算效率上出现严重缺陷，例如，要在一个存放有 1000 万张信用卡客户数据的数据库中，遍历所有客户的联系人网络，几乎成为一项无法完成的任务。

好在面对困难时，总会出现转机，Spark Graphx 图计算引擎和 Neo4j 图数据库的出现，降低了关系网络的处理和应用门槛。本章使用 Spark Graphx 和 Neo4j 来处理关系网络问题，它将带给读者完全不同的畅快体验。

## 15.1 社会网络和属性图

社会网络可以使用属性图来描述。属性图由英文 property graph 翻译而来，它是一个有向多重图，顶点和边上均可附带相关信息作为其属性 (properties)。

图 15-1 为一个属性示意图，由顶点（vertex）和边（edge）构成，边表示两个顶点之间存在某种关系。图 15-1 中的属性图共有 5 个顶点，每个顶点有一个“姓名”属性，小王、小李、老陈、老张、老刘分别为对应顶点的属性值。顶点之间的边可以有多个属性，图中所示的每条边即有两个属性：借款日期和借款金额，例如小王和老张之间的边，表示老张在“2015/02/10”向小王借了“¥80000”。

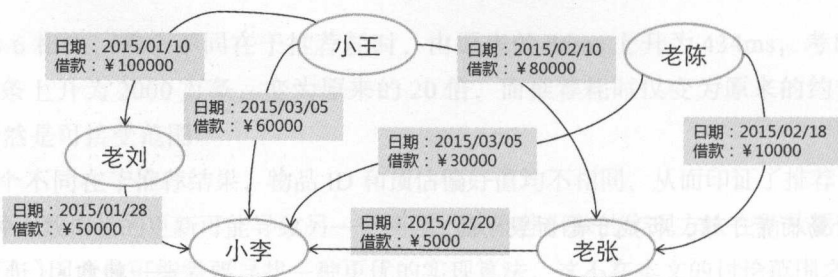


图 15-1 属性图示例

可以想象,利用属性图可以非常自然地描述社会网络。社会网络中的实体由属性图的顶点来表示,实体本身的属性即为顶点的属性(如姓名、年龄等),实体之间的关系由边来表示,关系的属性即为边的属性(如借款日期、借款金额等)。

生活和工作中充斥着各种各样的社会（关系）网络，它们都是各种各样的属性图。从属性图顶点的类型来看，可以将关系网络分为单一类型顶点的关系网络和多类型顶点的关系网络。单一类型顶点的关系网络中的所有顶点，都是同一种类型的实体，例如图 15-1 中的顶点全部是人。再如银行转账记录形成的关系网络，用户是对应属性图中的顶点，这些顶点全部是用户实体。

多类型顶点的关系网络对应的属性图中，存在两个以上的顶点类型。例如淘宝网购物记录形成的关系网络中，有购买者、店铺、商品三种类型的顶点，顶点类型越多，属性图越复杂，也越难处理。在实际应用中，一般将顶点类型限制在两种以内，如上述淘宝网购物记录的关系网，可以通过将商家作为商品的属性，将其简化为两种顶点类型的属性图，如图 15-2 所示。

在图 15-2 中，本来存在三种类型的顶点：购买者、店铺和商品，为了降低复杂度，可以将店铺看成商品的一个属性，这样就将原来的三种类型的顶点，简化为两种类型的顶点，简化后的属性图中仅包含购买者和商品两种顶点类型。当然，也可以将商品作为店铺的属



性，这取决于你如何关注待处理的关系网络。

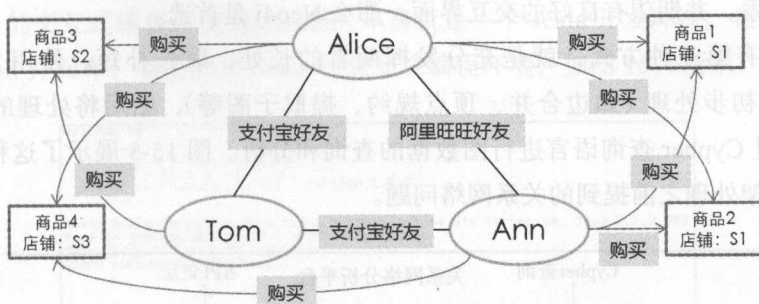


图 15-2 两种顶点类型的属性图

我们知道，关系网络广泛存在于我们的生产生活中，但由于关系网络的复杂性（顶点众多，关系庞杂），传统数据库对关系网络的处理显得力不从心。不管是 SQL 脚本复杂度还是计算性能上，使用 Oracle、DB2 等关系型数据库遍历关系网络是一件非常困难的事情。使用关系型数据库在 2000 万个信用卡客户中，搜寻并建立联系人关系网络几乎是一件无法实现的事情。

## 15.2 Spark GraphX 与 Neo4j

关系型数据库解决关系网络遇到的困难，在于关系网络的数据存储方式和计算方式。开源的 Spark GraphX（Spark 图计算）和 Neo4j 解决了属性图的存储和计算问题，是处理关系网络的利器之一。

Spark GraphX 是 Spark 提供的一个图计算引擎，由于 Spark 出身于大数据并行处理，作为组件之一且专注于图计算的 GraphX 能够提供良好分布式存储和并行处理；Neo4j 同样也可以对图进行存储、计算、查询和展示。两者同样基于属性图进行计算和处理，但就目前来看，两者仍然各有侧重，表 15-1 列举了两者的简单对比。

表 15-1 Spark GraphX 和 Neo4j 的对比

	Spark GraphX	Neo4j
图并行存储	专家	支持
图并行计算	专家	支持
图查询	无专门查询语言	Cypher 查询语言
图展示	无图形展示	有图形展示
支持的编程语言	Scala、Java、Python	Java

从表 15-1 的对比可以看出，如果面临的是大规模的关系网络之间属性的计算，那么 Spark GraphX 是首选。Spark GraphX 是由 Scala 语言编写的，能够与 Java 无缝转换，简洁

而强大的 Scala 语言让本来烦琐的图计算代码变得短小精悍。但是，如果你的需求是从图中发现特殊的关系，并期望有良好的交互界面，那么 Neo4j 是首选。

当然，还有第三种方式，就是充分发挥两者的长处，取长补短。使用 Spark Graphx 对图数据进行初步处理（如边合并、顶点规约、提取子图等），然后将处理的结果输出至 Neo4j，并通过 Cypher 查询语言进行图数据的查询和分析。图 15-3 展示了这种框架，我们将使用这种框架处理之前提到的关系网络问题。

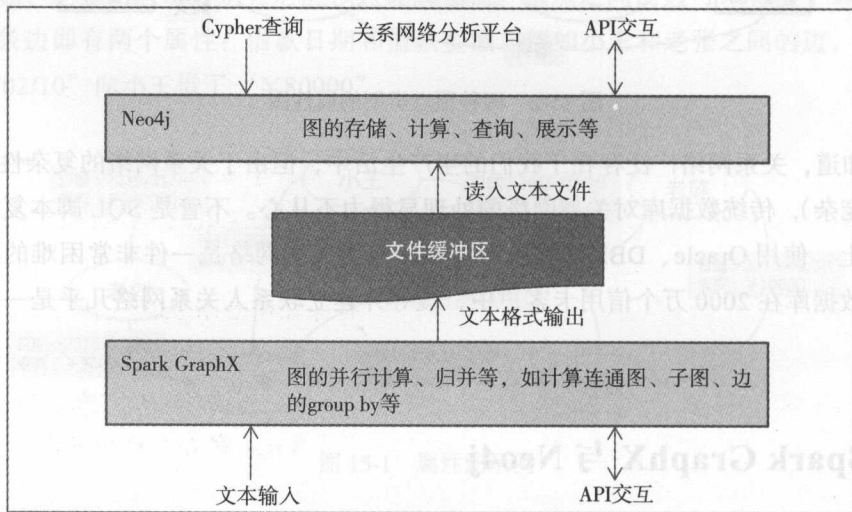


图 15-3 Spark GraphX 和 Neo4j 结合的图处理平台

当然，你可以结合具体的应用场景来决定是单独采用 Spark GraphX 或 Neo4j，还是将两者结合起来。本书使用 Spark GraphX 处理原始关系网络，并在原始关系网络的基础上通过图计算输出一个满足特定条件的结果网络（子网络），Neo4j 在 Spark GraphX 的处理结果（子网络）上进行交互式的查询分析。

在真正进行关系网络的处理之前，有必要简单介绍将要使用的 Scala 语言和 Cypher 图查询语言。

### 15.2.1 Scala 编程语言

Scala 是一种函数式风格的编程语言，即“高阶”函数可以使用其他函数作为输入，这种函数式编程语言使其在 Spark 中得到了广泛应用。与 Java 语言一样，Scala 语言也运行于 JVM 之上，并且很多人认为 Scala 是下一代的 Java 语言，所以尽管 Scala 代码与 Java 代码看起来一点都不一样，但是，如果你对 Java 编程有很深的理解，那么学习 Scala 语言将是一件非常轻松的事情。

Spark GraphX 是完全基于 Scala 语言的, 使用 Scala 进行 Spark 编程水到渠成, 并且 Spark GraphX 的交互式编程环境 spark-shell 本身就是一个 scala 开发环境。在 Spark 集群中输入 spark-shell 并回车, 就会启动 Scala 的交互式编程环境, 如图 15-4 所示。

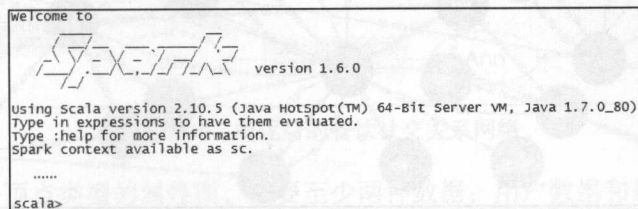


图 15-4 启动 spark-shell

从图 15-4 可以看到, spark-shell 启动后, 实际上是进入了 scala 交互界面。由此对于希望使用 Spark GraphX 进行开发的读者, 需要具备 Scala 的基本知识, 并且需要参照 Spark GraphX 官方文档掌握 Scala 在 Spark GraphX 中的使用方式。

本书的目的不在于介绍 Scala 语言本身, 也不会教你如何使用 Scala 语言, 但是, 如果希望读懂本章随后的内容, 具备基本的 Scala 语言基础和 Spark 编程基础是必须的。

你不需要专门去学习 Scala 语言, 一条“捷径”就是: 阅读 Spark 官方文档详细了解 Spark 和 Spark GraphX 编程方法 (这是必须的), 在阅读 Spark 官方文档的过程中, 你会慢慢对 Scala 有所了解。

## 15.2.2 Cypher 查询语言

使用 Neo4j 操作图数据库, 如果要使用交互式查询, 那么要使用 Cypher 查询语言。这是一门类似 SQL 的查询语言, 如果你熟悉 SQL, 那么学习 Cypher 语言也不会有问题。

Cypher 语言是 Neo4j 中的 SQL, 它是一门描述性的图形查询语言, 允许不必编写图形结构的遍历代码对图形存储有表现力和效率的查询。初次接触 Cypher 这种描述式的查询语言, 你可能会稍感“吃惊”, 因为它用非常直观的方式表示图的结构, 比如, 使用 (a) → (b) 表示顶点 a 到顶点 b 之间有一条边。

Cypher 语言可以方便查询 Neo4j 图数据库中的最短路径、指定深度的路径等关系, 并且用直观的图形展示出来 (当然也可以输出成表格形式)。例如, 图 15-5 即是通过 Cypher 查询输出的结果。

Neo4j 官网上有 Cypher 的完整帮助文档, 读者可以阅读官方文档对 Cypher 查询语言进行初步了解。本书随后内容并不需要复杂的 Cypher 查询, 基本了解 Cypher 语言的语法即可读懂本章内容, 但如果需要工作中熟练使用 Neo4j, 则需要花点时间深入学习该查询语言。

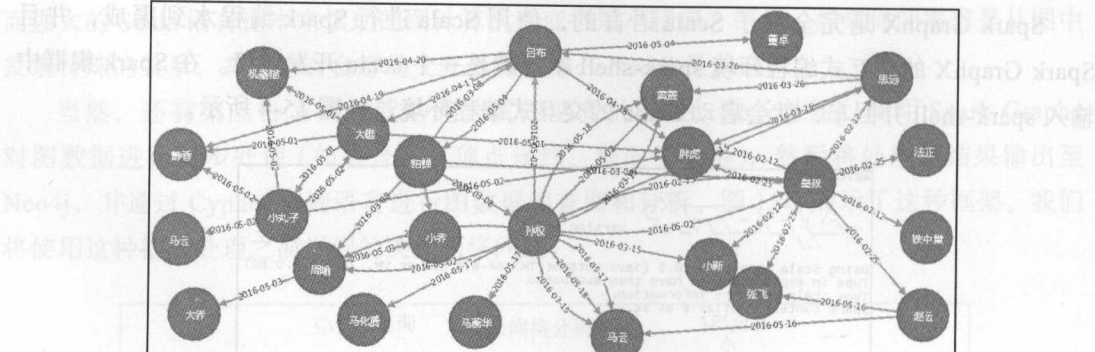


图 15-5 Cypher 的图形输出结果

## 15.3 使用 Spark GraphX 和 Neo4j 处理社会网络

### 15.3.1 背景说明

公司推出了一款餐饮社交 APP，为用户提供好友聚餐的线上平台，每个用户都有自己的“吃货好友群”，可以请自己的吃货好友吃饭，也可以接受好友的邀请。

另外，根据六度分割理论（six degrees of separation），在这个社会里，任何两个人之间建立一种联系，最多需要六个人（包括这两个人在内），无论这两个人是否认识，都可生活在地球的任何一个地方。因此，该 APP 还可以提供你认识新朋友的机会，可以让好友的好友成为自己的好友。

该 APP 会记录下每个用户与吃货好友之间的用餐关系，包括时间、餐厅、金额等。这些信息构成一种典型的社会关系网络。我们将使用 Spark Graphx 和 Neo4j 对该社会关系网络进行处理。

### 15.3.2 数据准备

从 15.3.1 节可以知道, 该餐饮社交 APP 记录的数据包括用户信息、吃货好友间的聚餐信息、用餐的餐厅信息等。这些信息组成的是一个多顶点类型的属性图, APP 用户和餐厅是两种实体, 对应了两种顶点类型。

使用 15.1 节的方法，我们将餐厅视为聚餐关系的一个属性（包含在订单信息中），就可以将上述两种顶点类型的属性图简化为单一顶点类型的属性图，简化后的属性图只需要用户这一个顶点类型，如图 15-6 所示。



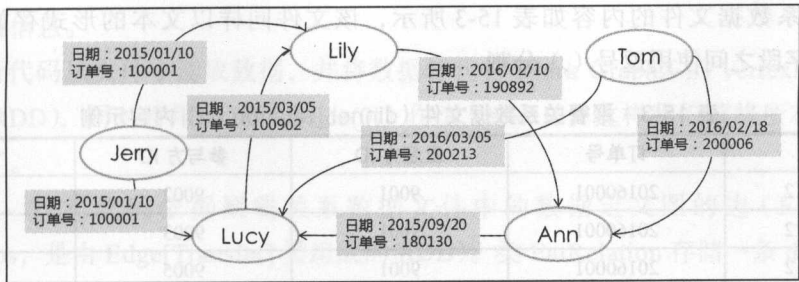


图 15-6 简化后的餐饮社交关系网络

构建上述单一顶点类型的属性图，需要至少两种数据：用户数据和聚餐关系数据（订单数据是聚餐关系的一个属性，为了简便起见，本文中不包含订单数据）。这两部分数据需要从餐饮社交 APP 的后台数据库中导出，并通过 ETL 过程转化为关系网络期望的格式（本章采用文本数据）。就 Spark Graphx 和 Neo4j 的要求来说，用户数据（假设 ETL 得到的文件名称为 user.txt）和聚餐关系数据（ETL 得到的文件名称为 dinner\_relation.txt）字段要求如图 15-7 所示。

user	dinner_relation
用户ID	日期
用户昵称	订单号
用户描述	饭局发起方ID
	饭局参与方ID
	备注

图 15-7 用户数据和聚餐关系数据

为了方便构建聚餐关系，我们将 APP 后台数据库中的数据经过 ETL 过程加工成图 15-7 所示的格式。每种聚餐关系（dinner\_relation）有一个唯一的订单号，并且每种聚餐关系包含一个发起方，其余的均为饭局参与方。例如，如果你请吃饭，那么你作为发起方，其他人均参与方；如果 AA 制，则可任选其一作为发起方。

15.3.3 Spark GraphX 处理原始网络

现在，我们具体看看 ETL 得到的数据文件的内容，首先是用户数据，其内容如表 15-2 所示，字段之间使用制表符（\t）分割。

表 15-2 用户数据文件（user.txt）内容示例

用户 ID	用户昵称	用户描述
9001	皇叔	蜀国老大
9002	胖虎	其实不胖
9003	小新	是个女的
9004	思远	想的远

聚餐关系数据文件的内容如表 15-3 所示，该文件同样以文本的形式存放于 Hadoop HDFS 中，字段之间使用逗号 (,) 分割。

表 15-3 聚餐关系数据文件 (dinner\_relation.txt) 内容示例

日期	订单号	发起方 ID	参与方 ID	备注
2016-02-12	20160001	9001	9002	请吃饭
2016-02-12	20160001	9001	9003	请吃饭
2016-02-12	20160001	9001	9005	请吃饭
2016-02-12	20160001	9001	9004	请吃饭
2016-02-25	20160002	9001	9002	AA

为了便于交互，我们采用 spark-shell 方式进行数据处理，你也可以使用 scala 项目的形式进行数据处理。

启动 spark-shell，依次输入以下 Scala 代码（可以将以下 Scala 代码保存到一个文件中，编译后一次性执行）。首先通过用户数据文件和聚餐关系数据文件分别构建 GraphX 的顶点和边，如代码清单 15-1 所示。

代码清单 15-1

```

1. import org.apache.spark._
2. import org.apache.spark.graphx._
3. import org.apache.spark.rdd.RDD
4. // 定义一个 User 类来存储用户信息：ID、姓名、描述
5. case class User(userId: Long, userName: String, userDesc: String)
6. // 从文件中构建用户 RDD，同时将用户 ID 作为 VertexId
7. val users: RDD[(VertexId, User)] =
8.   sc.textFile("/home/datamart/graphx/eat/user.txt").map{ line =>
9.     val fields = line.split("\t")
10.    (fields(0).toLong, User(fields(0).toLong, fields(1), fields(2)))
11.  }
12. // 定义一个 EatRelation 类，存储吃饭关系：请客方、被请方、日期、订单 ID 等
13. case class EatRelation(srcUserId: Long, dstUserId: Long,
14.                         trxDte: String, dinnerId: Long, eatType: String)
15. // 从文件中构建转账关系 RDD
16. val relationships: RDD[Edge[EatRelation]] =
17.   sc.textFile("/home/datamart/graphx/eat/dinner_relation.txt").map{ line =>
18.     val fields = line.split("\t")
19.     Edge(fields(2).toLong, fields(3).toLong,
20.          EatRelation(fields(2).toLong, fields(3).toLong, fields(0), fields(1).
21.            toLong, fields(4)))
22.  }
23. // 构建 graph
24. val graph: Graph[User, EatRelation] = Graph(users, relationships)

```

第 1 ~ 3 行代码导入所需的 Scala 包，第 5 行代码创建一个 case class，用于存储每个

用户的相关信息。

第 7 行代码从文件中读取数据，并将数据转化为 Spark GraphX 的 VertexRDD（存储顶点信息的 RDD），同时将用户 ID（fields(0)）作为 VertexId。这样我们就将所有的顶点信息定义完成了。

第 12 ~ 21 行代码依据聚餐关系数据文件中的数据定义图的边（Edge，即变量 relationships，是由 Edge[Transfer] 类组成的 RDD）。类 EatRelation 存储一条 dinner\_relation 信息。

现在，组成 Graph 的顶点和边都定义完成了，第 23 行代码完成了整个 Graph 的构建（变量名称为 graph）。需要注意的是，到目前为止，Spark GraphX 仅完成了图的定义，代码并未真正将文件读入内存中，Spark 只在需要的时候真正读入数据，这在大量数据存在的情况下是非常必要的。

通过代码，我们来查看该 graph 的顶点（graph.vertices）和边（graph.edges），分别如图 15-8 和图 15-9 所示。

```
scala> graph.vertices.foreach(println)
(9003,User(9003,小新,是个女的))
(9019,User(9019,马云,快递员))
(9013,User(9013,孙悦,美国老大))
(9021,User(9021,大乔,美女3))
(9017,User(9017,机器猫,哆啦A梦))
(9009,User(9009,貂蝉,美女1))
(9011,User(9011,吕布,战神2))
(9007,User(9007,张飞,武士))
(9015,User(9015,大雄,其实不雄))
(9005,User(9005,袁中兴,江湖人物))
(9001,User(9001,皇叔,蜀国老大))
(9023,User(9023,马化腾,腾讯老大))
(9002,User(9002,胖虎,其实不胖))
(9016,User(9016,静香,日本MM))
(9006,User(9006,法正,智者))
(9008,User(9008,赵云,战神))
(9010,User(9010,小乔,美女2))
(9014,User(9014,马云,阿里老大))
(9024,User(9024,马蔚华,打酱油))
(9004,User(9004,思远,想的远))
(9020,User(9020,黄盖,大将))
(9012,User(9012,周瑜,都督))
(9018,User(9018,小丸子,日本MM2))
(9022,User(9022,董卓,董卓老大))
```

图 15-8 graph 的顶点信息

```
scala> graph.edges.foreach(println)
Edge(9008,9007,EatRelation(9008,9007,2016-05-16,20160012,请吃饭))
Edge(9008,9014,EatRelation(9008,9014,2016-05-16,20160012,请吃饭))
Edge(9011,9009,EatRelation(9011,9009,2016-05-04,20160011,请吃饭))
Edge(9011,9022,EatRelation(9011,9022,2016-05-04,20160011,请吃饭))
Edge(9012,9010,EatRelation(9012,9010,2016-05-03,20160010,请吃饭))
Edge(9012,9021,EatRelation(9012,9021,2016-05-03,20160010,请吃饭))
Edge(9013,9001,EatRelation(9013,9001,2016-05-02,20160008,请吃饭))
Edge(9013,9011,EatRelation(9013,9011,2016-05-02,20160008,请吃饭))
Edge(9013,9012,EatRelation(9013,9012,2016-05-02,20160008,请吃饭))
Edge(9013,9012,EatRelation(9013,9012,2016-05-18,20160014,AA))
Edge(9013,9014,EatRelation(9013,9014,2016-05-17,20160013,请吃饭))
Edge(9013,9014,EatRelation(9013,9014,2016-05-18,20160014,AA))
```

图 15-9 graph 的边信息

现在 graph 包含由所有用户聚餐数据构成的属性图，假设我们要分析的是 2016 年的数据，那么需要从 graph 中选取 2016 年的子图，如代码清单 15-2 所示。

## 代码清单 15-2

```

1. val subGraph = graph.subgraph(
2.     epred = (triplet) => {
3.         val trxDte = triplet.attr.trxDte
4.         trxDte.substring(0,4) == "2016"
5.     }
6. ).partitionBy(PartitionStrategy.EdgePartition2D)

```

使用 Graph 类提供的 subgraph 函数可以将日期前 4 位为“2016”的子图筛选出来。第 29 行代码将得到的子图重新分区，其目的是在随后对边进行 group 的计算中能够得到正确的结果（请查阅 Spark GraphX 的 API 接口文档）。

到目前为止，我们已经从整个庞大的关系网络中得到了日期为 2016 年的子图，但这个子图顶点之间可能存在同方向的多条边，看起来可能如图 15-10 所示。

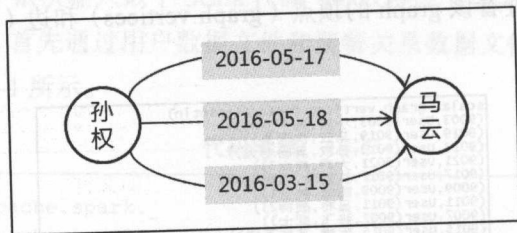


图 15-10 subGraph 的结构示意图

而我们最终需要的图结构是图 15-11 所示的样子，边的属性值“3”表示用户“孙权”在 2016 年请用户“马云”吃了 3 次饭。

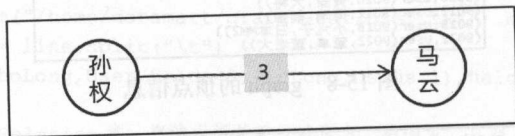


图 15-11 边归并后的结构示意图

为了得到图 15-11 所示的属性图，我们还需要对相同起始顶点的边进行归并（group），同时计算吃饭次数，如代码清单 15-3 所示。

## 代码清单 15-3

```

1. // groupby Edges, 合并边，计算边的个数
2. val subGraphUniqEdge = subGraph.mapEdges{
3.     edge => 1
4. }.groupEdges{
5.     (e1,e2) => e1+e2
6. }

```



上述第 32 行代码将每条边视为 1 次请吃饭记录, 并通过第 34 行代码将所有的同向边条数累加起来, 最终得到两个用户之间的同向边条数。

经过上述过程, 我们已经从庞大而复杂的关系图中获得了一个经过归并的子图, 这已经满足了本章所要的示例效果, Spark GraphX 的计算到此结束, 接下来将计算好的子图输出为文本数据, 供随后的 Neo4j 使用。子图输出的代码如代码清单 15-4 所示。

代码清单 15-4

```
1. subGraphUniqEdge.edges.map{ edge =>
2.   Array(edge.srcId.toString, edge.dstId.toString, edge.attr.toString).mkString(",")
3. }.saveAsTextFile("/data/graphx/eat/output/")
```

上述第 38 行代码将自动创建 “/data/graphx/eat/output/” 目录, 并在该目录下写入数据。图 15-12 所示即为该文件夹对应的内容, part-00000 和 part-00001 是真正的数据文件, \_SUCCESS 是一个标记文件, 用于标识结果是否成功输出。

```
total 8
-rw-r--r-- 1 datamart datamart 63 Mar  3 14:41 part-00000
-rw-r--r-- 1 datamart datamart 364 Mar  3 14:41 part-00001
-rw-r--r-- 1 datamart datamart  0 Mar  3 14:41 _SUCCESS
```

图 15-12 结果图输出目录中的文件

查看上述结果文件, 会看到图 15-13 所示的内容。下面将使用上述结果文件, 以及之前使用的用户数据文件 (user.txt) 构建 Neo4j 中的图。

```
[datamart@datamart2 output]$ cat part-0000*
9004,9002,1
9004,9011,1
9004,9020,1
9008,9007,1
9008,9014,1
9012,9010,1
9012,9021,1
9018,9002,1
9018,9015,1
9018,9016,1
9018,9017,1
9018,9019,1
9001,9002,2
```

图 15-13 结果文件中的数据

### 15.3.4 Neo4j 交互式查询分析

总算到 Neo4j 出场了! 既然 Neo4j 是一个图数据库, 与关系数据库一样, 首先需要创建一个数据库, 出乎意料的是, 这个过程出奇的简单! 你所要做的就是到 Neo4j 官网下载 Neo4j 的社区版 (Windows 和 Linux 均有对应的安装介质), 然后进行安装。

安装完成后, 启动 Neo4j (见图 15-14)。此时你只需要指定一个目录, 单击 “Start” 按钮, Neo4j 会自动在对应目录中创建图数据相关的配置文件, 稍等片刻, Neo4j 数据库服务即启动完成。

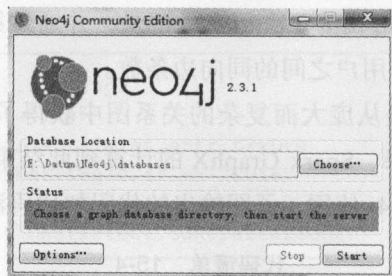


图 15-14 Neo4j 首次启动界面

随后在浏览器中输入 <http://localhost:7474>，可以进入 Neo4j 的 Cypher 交互页面，如图 15-15 所示。

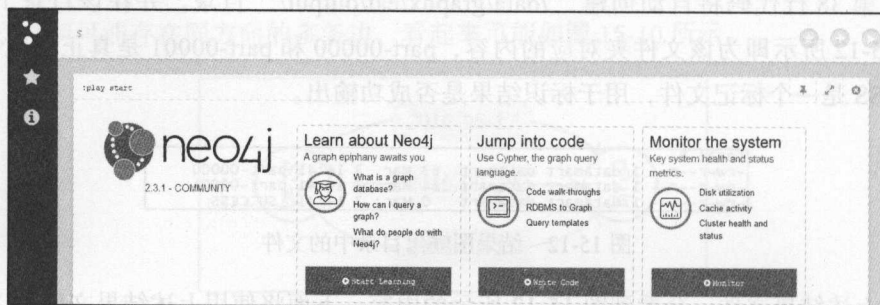


图 15-15 Cypher 交互界面

接下来，我们将上一节的输出结果连同用户数据文件（user.txt）导入 Neo4j 数据库中。在图 15-15 所示的输入框（\$ 提示符之后）中输入如代码清单 15-5 所示的代码。

代码清单 15-5

---

```

1. -- 导入用户列表
2. LOAD CSV FROM "file:///E:/Datum/Spark/Graphx/吃饭关系网/user.txt" AS line
   FIELDTERMINATOR '\t'
3. CREATE (:User { userId: toInt(line[0]), userName: line[1], userDesc:
   line[2]})

```

---

上述代码从 user.txt 中读入用户数据，并将每行数据对应至一个 User 节点（代码第 2 行）中。

用户顶点已经存储到 Neo4j 之中了，下面将 15.3.3 节中得到的结果数据文件（part-00000 和 part-00001）加入数据库，如代码清单 15-6 所示。

代码清单 15-6

---

```

1. LOAD CSV FROM "file:///E:/Datum/Spark/Graphx/吃饭关系网/output/part-00000" AS row

```

---

```

2. FIELDTERMINATOR ','
3. MATCH (src:User), (dst:User)
4. WHERE src.userId = toInt(row[0]) AND dst.userId = toInt(row[1])
5. CREATE (src)-[ et: eatCount {count:toInt(row[2])} ]->(dst)
6. LOAD CSV FROM "file:///E:/Datum/Spark/Graphx/吃饭关系网/output/part-00001" AS row

```

现在，一个完整的关系图已经存放到 Neo4j 之中了。下面就可以通过 Cypher 语言进行关系网络的查询分析。

首先，在输入框中输入代码 “MATCH(a:User) RETURN a”，查看整个关系网络的全貌，如图 15-16 所示（注意，如果你的关系网络非常大，请不要这么做，本书出于演示目的，仅导入了少量数据）。

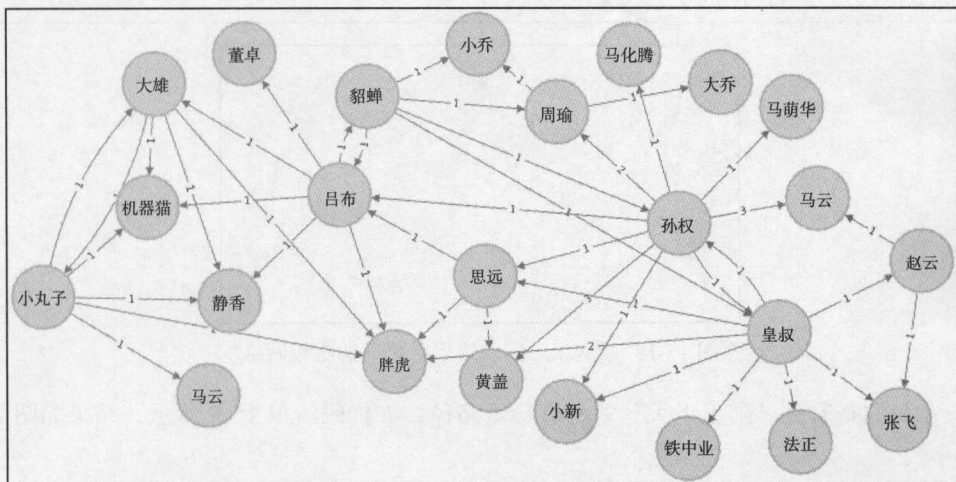


图 15-16 计算后的吃饭网络全貌

输入代码清单 15-7 查看“皇叔”到“马云”的关系网络，结果如图 15-17 所示。

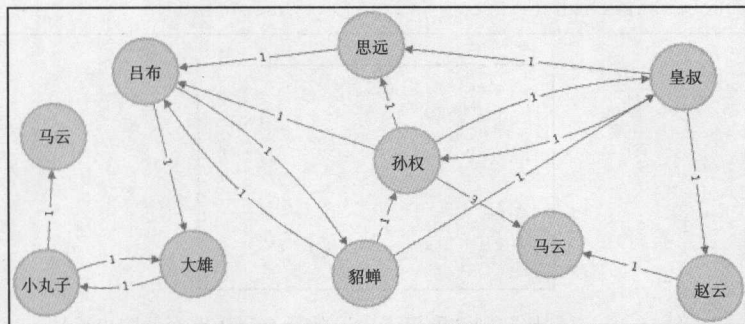


图 15-17 “皇叔”到“马云”的吃饭网络

代码清单 15-7

```

1. match (a:User),(b:User)
2. where a.userName=" 皇叔 " and b.userName=" 马云 "
3. return (a)-[*..10]->(b)

```

可以看到,图 15-17 中出现了两个“马云”,我们希望找到描述是“阿里老大”的用户,如代码清单 15-8 所示,结果如图 15-18 所示。

代码清单 15-8

```

match (a:User),(b:User)
where a.userName=" 皇叔 " and b.userName=" 马云 " and b.userDesc=" 阿里老大 "
return (a)-[*..10]->(b)

```

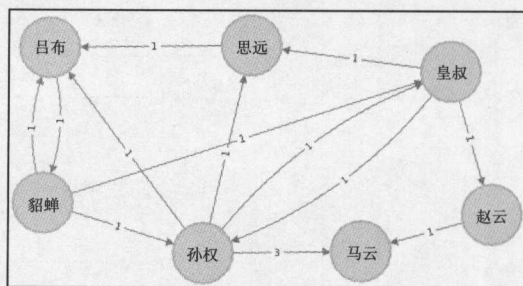


图 15-18 “皇叔”到“阿里老大”的吃饭网络

现在我们查看到“阿里老大”的所有最短路径,如代码清单 15-9 所示,结果如图 15-19 所示。

代码清单 15-9

```

match (a:User),(b:User)
where a.userName=" 皇叔 " and b.userName=" 马云 " and b.userDesc=" 阿里老大 "
return allShortestPaths( (a:User)-[*..10]-(b:User) )

```

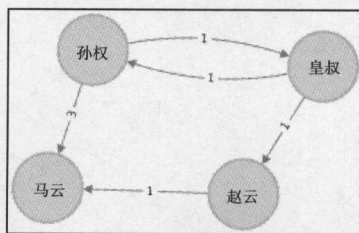


图 15-19 “皇叔”到“阿里老大”的所有最短路径（按步长）

到目前为止,我们已经通过 Neo4j 的 Cypher 查询语言查看了聚餐关系网络中的一些有



趣的关系,通过深入学习 Spark Graphx 和 Neo4j,读者可以发现更多有趣且实用的场景。

### 15.3.5 更多的应用场景

#### 1. 信用卡催收

当持卡人还款发生延期的时候,将会进入催收程序。催收需要联系持卡人以及持卡人的联系人,因此通过持卡人提交的申请资料(填写的联系人手机号),以及通话记录数据(可与第三方合作获得)等,绘制持卡人-联系人关系网络可以为催收提供重要的数据支撑。

例如,图 15-20 所示的为某联系人关系网,边上的数字表示近半年的通话记录数,根据边的属性,可以找到最亲密的联系人(假定通话次数越多,表示关系越亲密)。

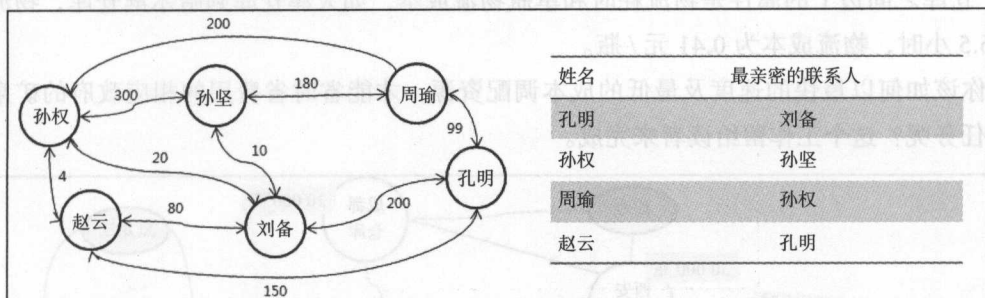


图 15-20 最亲密的联系人

#### 2. IT 服务架构管理

关系网络可以应用于公司 IT 服务器架构管理,公司众多的 IT 系统服务模块可以视为属性图的顶点,服务模块之间的调用关系则形成了属性图的边。图 15-21 所示的为简单的服务模块调用关系网。

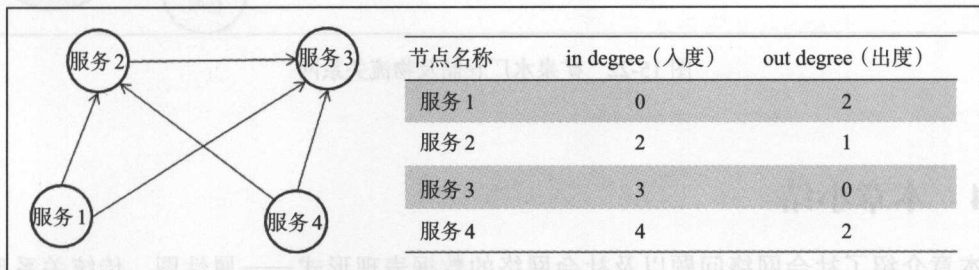


图 15-21 服务模块调用关系网

调用关系使用一个有向边,从调用者出发指向被调用者,在属性图中,指向顶点的边的条数称为该顶点的入度(in degree),离开顶点的边的条数称为该顶点的出度(out degree)。因此,我们可以用顶点的入度来衡量该顶点在调用关系网中的重要程度,比如图

15-21。服务 3 的入度为 3，是所有顶点中最大的，因此服务 3 是该调用关系中的关键节点，如果服务 3 停止，那么影响范围最大。

### 3. 仓储及物流管理

仓储和物流也构成了一个关系网络，通过关系网络的管理可以方便寻找出最优路径和最经济的配送方案。

笔者在读大学期间，曾经发生过由于上游某化工厂事故导致松花江大面积污染事件，影响到哈尔滨及下游众多城市的饮水安全。当时超市中的矿泉水被抢购一空，如果你是一家矿泉水厂厂长，图 15-22 是你面对的仓储和物流关系网，存在两种顶点类型：仓库（圆形顶点）和销售区（椭圆形顶点），仓库顶点的属性是库存量（例如成都仓库库存量是 20000 瓶），仓库之间边上的属性是物流耗时和单瓶物流成本，如天津仓库到哈尔滨仓库，物流耗时 16.5 小时，物流成本为 0.41 元/瓶。

你该如何以最快的速度及最低的成本调配资源，才能省时省费用的相应政府的矿泉水调配任务呢？这个工作留给读者来完成。

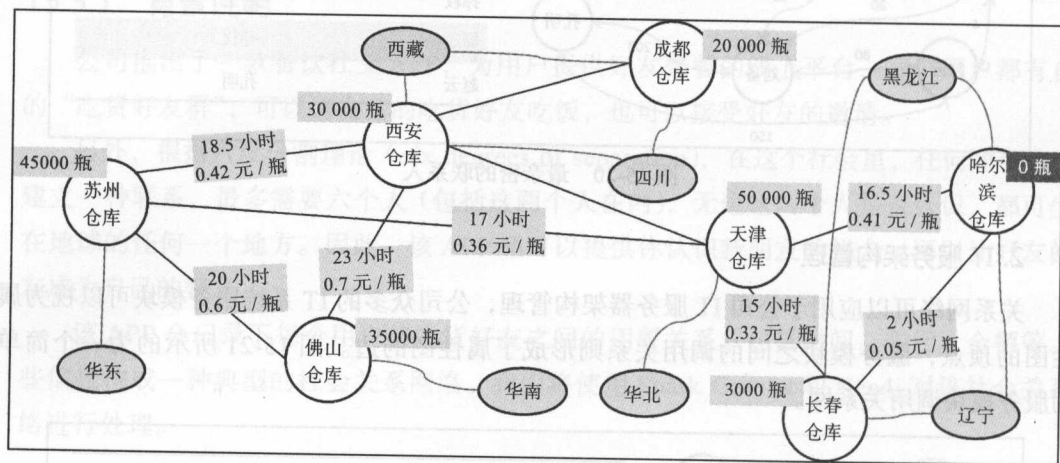


图 15-22 矿泉水厂仓储及物流关系网

## 15.4 本章小结

本章介绍了社会网络问题以及社会网络的数据表现形式——属性图，传统关系型数据库对属性图的存储和计算都存在一定的困难，而针对图计算和存储的 Spark Graphx 和 Neo4j 可以轻松解决属性图的计算、存储和展示问题。

Scala 语言是使用 Spark Graphx 的首选，对于使用过 Java 的读者，Scala 应该不是问题。对于没有接触过 Java 和 Hadoop 编程的读者，需要花些时间了解 Scala 的基本语法，当然，

Spark Graphx 也支持 Python 语言,但至本书成书时,Python 语言仅支持部分 Graphx API。

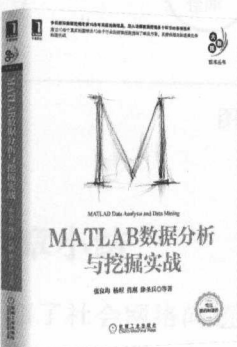
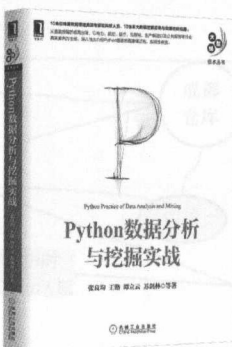
Neo4j 的查询语言 Cypher 是一种类 SQL 的查询语言,简单易学,如果希望从 Neo4j 查看网络的图形展示,读者需要简单学习 Cypher 语言。

本章以餐饮社交关系网络为例,介绍了 Spark Graphx 结合 Neo4j 进行属性图处理的一种方式,并介绍了一些社会网络的应用场景。读者可以根据自己的实际工作要求发现更多的应用场景,而图计算则是解决这些关系网络应用场景的新的手段。

新的技术可以让以前难以处理的问题迎刃而解,但技术仍然是一种手段,如何让技术解决问题、让数据产生价值是数据科学家们应该思考的问题。

## 推荐阅读

### 大数据学习路线图：数据分析与挖掘





## 作者简介

**陈春宝** 上海交通大学工业工程博士，经济学硕士。在银行、信用卡、医药与电信等行业拥有近10年数据挖掘分析与SAS建模经验，现就职于商业银行，在数据挖掘、机器学习和业务咨询方面有着独到的见解。工作跨大数据、营销、风险、运营等多个领域，擅长诊断各类业务问题，应用商业和数据分析手段获得创新性的解决方案，并帮助业务部门落地。曾经担任MSA咨询顾问、交通银行信用卡中心数据分析经理、上海交通大学工程硕士企业导师。在SCI&EI索引期刊发表论文10余篇。

**阙子扬** 哈尔滨工业大学计算机硕士。在银行、信用卡、金融服务行业拥有9年系统开发和数据应用经验。现任某金融服务公司创新产品线研发部经理，负责大数据产品创新和系统开发工作。先后任职于中国平安、交通银行信用卡中心、招商银行信用卡中心。对数据平台建设和数据系统开发应用有深刻理解，主导完成的标签系统、数据自助营销平台等项目取得了显著效果，使业务人员的营销效率由数周简化为数小时。

**钟飞** 华东师范大学统计学硕士。有近十年数据分析与建模经验，专注于信用卡、电子商务等行业的风控、运营与营销领域。现就职于某知名金融服务集团有限公司，负责电子商务平台风险策略分析工作。曾任eBay大中华区及交通银行信用卡中心数据分析师。能灵活运用主流数据分析工具，精通SAS、SQL、SPSS、R等。生活中爱好跑步，参加过多次半程马拉松，也造就了个人踏实、坚韧、靠谱的品性。

DT时代，大数据挖掘与分析将成为关系国计民生的新型技术，为各行各业带来新的发展动力。本书立足企业实践，对大数据平台、分析建模与系统应用等数据科学的各个方面做了系统全面的介绍，兼具理论与实战，对从事大数据分析的专业人员和企业信息系统建设工作都有参考价值。

—— **谢华美** 中国人民银行征信中心数据部负责人，数据分析专家

关于大数据技术和理论的书很多，但怎么应用数据解决现实的业务问题，恐怕最能给出答案的还是实际的数据从业者。三位作者基于近十年的从业经验，结合案例介绍数据挖掘在企业的实施过程，对企业人员能够提供直接帮助，也可帮助在校学生拓展视野，这些将让本书从同类书中脱颖而出。

—— **王丽亚** 上海交通大学工业工程系教授，博士生导师

如何从数据中提炼尽可能多的信息和知识，驱动商业模式变革进而创造新的市场竞争力，正是数据挖掘与分析技术的核心价值。本书从企业实战角度，介绍数据从线下分析到线上应用的具体过程，值得业务部门和数据分析部门借鉴。

—— **陈治龙** 浦发银行拉萨分行信息科技部总经理

任何一家希望长久发展的公司都必须重视数据的获取、治理和运用，通过数据挖掘、机器学习、人工智能等算法从数据中攫取新的洞察力，创新商业模式、优化业务流程或打造新的核心竞争力。本书内容覆盖企业内大数据流转的主要环节，并通过翔实的案例介绍常用数据挖掘、机器学习等算法及其建模过程，对企业来说是一本最佳的实践操作指导手册。

—— **冯炯** 嘉银（你我贷）副总裁，前阿里资深技术专家

“大众创业、万众创新”，经济增长点在改变，金融也在持续改革，获取更全面的客户数据，掌握从海量数据中挖掘知识宝藏、攫取洞察力的技术，是各类企业开拓金融新业态的关键。作者依托多年业务经验和专业知识，介绍了大量数据科学和分析技术的实际应用。对于那些力图将大数据转化为经营决策动力的企业，本书是最适合的操作指引。

—— **冯超** 泛合金融咖啡联合创始人/甲骨文（中国）大数据资深顾问



上架指导：计算机/大数据

ISBN 978-7-111-55680-0



定价：69.00元

投稿热线：(010) 88379604  
客服热线：(010) 88379426 88361066  
购书热线：(010) 68326294 88379649 68995259

华章网站：www.hzbook.com  
网上购书：www.china-pub.com  
数字阅读：www.hzmedia.com.cn